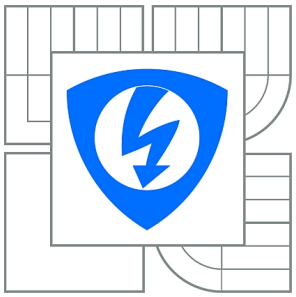
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

BEZOBSLUŽNÁ NABÍJECÍ STANICE PRO ELEKTROMOBILY

SELF-SERVICE STATION FOR CHARGING OF ELECTRIC AUTOMOBILES

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

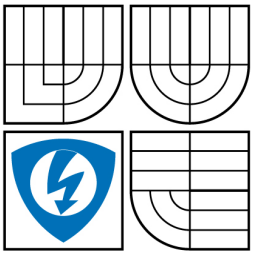
AUTOR PRÁCE — Bc. JAN KUBIZŇÁK

AUTHOR

VEDOUCÍ PRÁCE — Ing. PAVEL KUČERA, Ph.D.

SUPERVISOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student:	Bc. Jan Kubizňák	***ID:***	78550
Ročník:	2	***Akademický rok:***	2009/2010

NÁZEV TÉMATU:

Bezobslužná nabíjecí stanice pro elektromobily

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte možnosti ovládání externích silnoproudých zařízení sloužících k nabíjení elektromobilů s možností měření spotřeby elektrické energie.
Realizujte HW uspořádání bezobslužné nabíjecí stanice za pomoci vývojového kitu DK-LM3S9B96. Pro tento HW navrhněte řídicí aplikaci pod OS FreeRTOS.
Porovnejte vámi vytvořený systém s některým z komerčně dostupných řešení.

DOPORUČENÁ LITERATURA:

[1] Barry, R. FreeRTOS Reference Manual - API Functions and Configuration Options [online]. [cit.2009-11-30]. Dostupné z <http://www.freertos.org/Documentation/FreeRTOS-documentation-and-book.html>.
[2] Texas Instruments. LM3S9B96 Microcontroller Datasheet [online]. Poslední revize 9.10.2009 [cit. 2009-11-22]. Dostupné z <http://www.luminarymicro.com/products/lm3s9b96.html>.

Termín zadání:	8.2.2010	***Termín odevzdání:***	24.5.2010

Vedoucí práce: Ing. Pavel Kučera, Ph.D.

prof. Ing. Pavel Jura, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Obsahem této diplomové práce je návrh a realizace softwarového vybavení pro bezobslužnou nabíjecí stanici společnosti ELNICO s.r.o. Práce obsahuje přehledovou část, kde je uvedena charakteristika nabíjecích systémů a některé výrobky dostupné na trhu. Dále jsou popsány komponenty systému, pro který je SW vyvíjen. Podrobněji jsou rozebrány zejména možnosti řídicího počítače, realizovaného vývojovým kitem DK-LM3S9B96, a operačního systému reálného času FreeRTOS. Tvorbou softwaru pro tento počítač se zabývá druhá - praktická část práce. Popsána je zde struktura programu a podrobně pak jednotlivé moduly, pomocí nichž jsou ovládány komponenty nabíjecí stanice. Funkčnost je demonstrována fotografiemi aplikace za běhu.

KLÍČOVÁ SLOVA

Elektromobil, nabíjecí systém, operační systém reálného času, FreeRTOS.

ABSTRACT

The subject of this diploma thesis is a creation of SW for the self-service charging station of ELNICO Ltd. company. The work consists of two main parts. The first part contains an overview of general charging stations for electro cars at current market. After this the prototype of the ELNICO system is described. It's main components and the control system realized by the development kit DK-LM3S9B96 are characterized here. The real-time operating system FreeRTOS, that is used by the kit, is also specified. The second part of the document deals with the SW design and realization. The thesis describes the structure and the main modules of the program. The functionality is then demonstrated by photos of running application.

KEYWORDS

Electric car, charging system, real-time operating system, FreeRTOS.

KUBIZŇÁK, J. Bezobslužná nabíjecí stanice pro elektromobily. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. 70 s. Vedoucí diplomové práce Ing. Pavel Kučera, Ph.D.

PROHLÁŠENÍ

„Prohlašuji, že svou diplomovou práci na téma „Bezobslužná nabíjecí stanice pro elektromobily“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.“

V Brně dne: **24.května 2010** . . . . . .

(podpis autora)

PODĚKOVÁNÍ

Rád bych poděkoval svému vedoucímu Ing. Pavlu Kučerovi, Ph.D. za průběžné vedení a cenné rady během celé doby tvorby této práce a firmě ELNICO s.r.o. za poskytnutí prostor a hardwarového i softwarového vybavení, a zároveň za některé důležité připomínky a financování celého projektu.

V Brně dne: **24.května 2010** . . . . . .

(podpis autora)

Obsah

Seznam obrázků

Seznam tabulek

1 ÚVOD

Vývoj automobilového průmyslu v poslední době naznačuje, že by se v blízké době na trhu mohla prosadit i jinak než tradičně poháněná vozidla - zejména elektromobily. Dokládá to i fakt, že prakticky každý velký výrobce automobilů ve svém portfóliu buď elektricky nebo alespoň hybridně poháněná vozidla má. Většímu rozmachu elektromobilů však prozatím brání mimo jiné absence systémů pro jejich nabíjení. Proto jsou realizovány projekty, které se touto problematikou zabývají.

Jedním z nich je mimo jiné bezobslužná nabíjecí stanice realizovaná společností ELNICO s.r.o. se sídlem ve Dvoře Králové nad Labem. V současné době je vyvíjen prototyp, na němž se podílí i tato diplomová práce. Jejím hlavním cílem je vytvořit softwarové vybavení, kterým bude umožněno nabíjecí stanici ovládat a monitorovat. K těmto účelům je určen řídicí počítač, jenž je při vývoji prototypu systému reprezentován vývojovým kitem DK-LM3S9B96 od firmy Texas Instruments.

Práce je členěna na části přehledovou a praktickou. V první z nich je uvedena charakteristika obecné nabíjecí stanice pro elektromobily doplněná o přehled některých výrobků na trhu. Následně je popsána struktura a komponenty systému, jenž je předmětem této práce. Charakterizován je také řídicí počítač a jím využívaný operační systém FreeRTOS. Přehledovou část završuje porovnání vyvíjeného systému s jedním z konkurenčních produktů.

Praktická část práce se zabývá návrhem a realizací řídicí aplikace pro nabíjecí stanici. Uvedena je struktura programu a především podrobný popis základních modulů, ze kterých se dílo skládá. Funkčnost je předvedena fotografiemi běžící aplikace s přiřazenými popisky ovládání. Nakonec je uveden kompletní přehled zdrojových modulů a jejich rozmístění v souborovém systému.

2 BEZOBSLUŽNÉ NABÍJECÍ STANICE PRO ELEKTROMOBILY

Náplní této kapitoly je definice základních vlastností bezobslužných nabíjecích stanic pro elektromobily a přehled některých výrobků dostupných na trhu.

2.1 Charakteristika nabíjecích stanic

Bezobslužné nabíjecí stanice pro elektromobily lze charakterizovat jako elektrická zařízení, která umožní zákazníkovi nabití baterie jeho elektromobilu bez přítomnosti obsluhy. Řidič po přijetí ke stanici připojí svůj vůz kabelem do zásuvky stanice a jednoduchým způsobem navolí, kolik energie chce dobít. Nabízí se několik způsobů zjištění požadavků od zákazníka - využít lze:

- tlačítkovou klávesnici,
- dotykový displej,
- mobilní telefon (zaslání požadavku ve tvaru SMS zprávy na speciální telefonní číslo).

Obdobně systém může informovat o své činnosti:

- zobrazením na displeji,
- zasláním SMS zprávy, která informuje o průběhu nabíjení, na mobilní telefon.

Další nutnou komponentou systému je platební systém. Použít lze:

- akceptor mincí,
- akceptor bankovek,
- čtečku platebních karet,
- čtečku čipových karet,
- mobilní telefon (placení opět např. SMS zprávou).

Akceptory mincí a bankovek jsou výhodné především kvůli jednoduchosti zapojení, vhodné jsou v podstatě pro jakékoli umístění. Nevýhodou je na druhou stranu nutná přítomnost zásobníku na mince nebo bankovky, kvůli kterému musí

stanice stejně nějaká obsluha kontrolovat a vybírat z ní přijaté peníze. Systém je pak určitě také z bezpečnostního hlediska více náchylný.

Při použití platební karty poslední dvě zmíněné nevýhody samozřejmě odpadají. Nevhodné však může být, že ne úplně každý zákazník platební kartu vlastní. Co se týče připojení čtečky k systému, řešení je rozhodně složitější než v případě akceptorů. Stanice musí umožňovat spojení s bankou, provozovatel sítě nabíjecích systémů musí mít s bankou určitou smlouvu.

Čtečky čipových karet jsou vhodné především pro stanice, které jsou umístěné např. na parkovištích firem nebo nájemních budov. Uživatel se identifikuje prostřednictvím své karty a spustí nabíjení. Platbu provádí buď dopředně (uložením peněz jako kreditu) nebo zpětně (fakturou).

Placení mobilním telefonem se jeví jako poměrně zajímavá možnost. Stanice může být umístěna na jakémkoli místě, kde je zároveň přítomen signál mobilních sítí, což je dnes prakticky všude. Nevýhodou může být určité časové zpoždění mezi odesláním požadavku a spuštěním nabíjení. Pro provozovatele to samozřejmě znamená nutnost uzavření smlouvy s některým mobilním operátorem.

2.2 Přehled trhu

Uvedeny jsou produkty, které byly vystavovány na veletrhu elektromobilů eCarTec 13. - 15. 10. 2009 v Mnichově a výrobky firmy Rittal, která v březnu 2010 vyhrála výběrové řízení společnosti ČEZ na rozmístění prvních nabíjecích stanic v ČR (viz. např. [9]).

2.2.1 EBG ComplEo

Systém ComplEo společnosti EBG (viz. obrázek 1) je určen pro nabíjení jednoho až dvou elektromobilů využitím svých 400 V zásuvek. Placení lze provádět buď RFID chipem nebo mobilním telefonem. Přes ten lze odeslat požadavek na telefonní číslo uvedené na malém displeji stanice, po úspěšné autorizaci je pak spuštěno nabíjení. Informace o průběhu nabíjení jsou zákazníkovi zasílány prostřednictvím krátkých textových zpráv.

Jako vhodná umístění terminálu jsou doporučována obchodní centra a veřejná či firemní parkoviště (z [21]).

Obrázek 1: ComplEo
(z [21])

2.2.2 Hectronic HecCharger

Nabíjecí stanice HecCharger společnosti Hectronic je koncipována k použití společně s parkovacím automatem CITEA stejného výrobce (oba systémy jsou znázorněny na obrázku 2). Zaměřen je tedy pro veřejná parkoviště. Je uváděno, že k systému CITEA lze připojit teoreticky neomezený počet stanic typu HecCharger, přičemž již bylo úspěšně testováno současné připojení 32 stanic. Komunikaci mezi systémy lze volit buď bezdrátovou nebo kabelovou. HecCharger může obsahovat 1-4 zásuvky různých typů: jednofázová 220-240 V, třífázová 380 - 415V, obojí s proudy až do 63 A (z [8]).

Co se týče parkovacího automatu CITEA, umožněno je dle konfigurace placení mincemi, bankovkami a kreditními kartami (z [19]).

Obrázek 2: HecCharger, CITEA
(z [8])

2.2.3 Schletter P.CHARGE

Společnost Schletter nabízí celou linii systémů P.CHARGE pro bezobslužné nabíjení elektromobilů (viz. obrázek 3). Standalone stanice (na obr. značena písmenem S) může být vybavena displejem, akceptorem mincí, systémem pro identifikaci přes RFID chip a systémem pro tisk dokladu o nabíjení. Počet a typ zásuvek je uváděn jako volitelný zákazníkem. Další výrobek (W) je určen pro montáž na zeď.

Při předpokládané větší koncentraci nabíjecích stanic (např. na parkovišti) je doporučováno použít centralizovaný systém s jedním masterem (CM) a větším počtem slaveů (CS, D). Master je designován pro použití s 4-řádkovým nebo touch displejem. Nabízí ethernet konektivitu, placení přes GSM modem nebo RFID chip a tisk dokladu (z [13]).

Obrázek 3: P.CHARGE
(z [13])

2.2.4 Nabíjecí stanice Rittal

Systémy firmy Rittal, které byly v březnu 2010 vybrány ve výběrovém řízení ČEZu na rozmístění prvních nabíjecích stanic v ČR, jsou charakteristické zejména vysokou modularitou. Nabízeny jsou tři základní koncepce:

1. Indoor (montovaná na zeď, určená k soukromému použití),
2. outdoor (stojan nebo montovaná na zeď, také k soukromému použití),
3. a outdoor s možností placení za odebranou energii (stojan, ke komerčnímu použití).

Stanice nabízí dvě nezávislá odběrná místa volitelně s napětím jednofázovým 230 V / 50 Hz nebo třífázovým 400 V / 50 Hz, nabíjecí proudy lze konfigurovat v rozsahu 16 - 63 A.

K identifikaci uživatelů systém obsahuje RFID čtečku karet určenou především pro čipové karty typu Mifare Classic. Nastavení nabíjení se provádí využitím dotykového grafického displeje, zároveň je umožněn vzdálený monitoring stanice díky integrovanému GSM/GPRS modemu.

Provozní teploty jsou udávány v rozsahu -20...+40 °C (vše z [15]). Ilustrační fotografie stanice je zobrazena na obrázku 4.

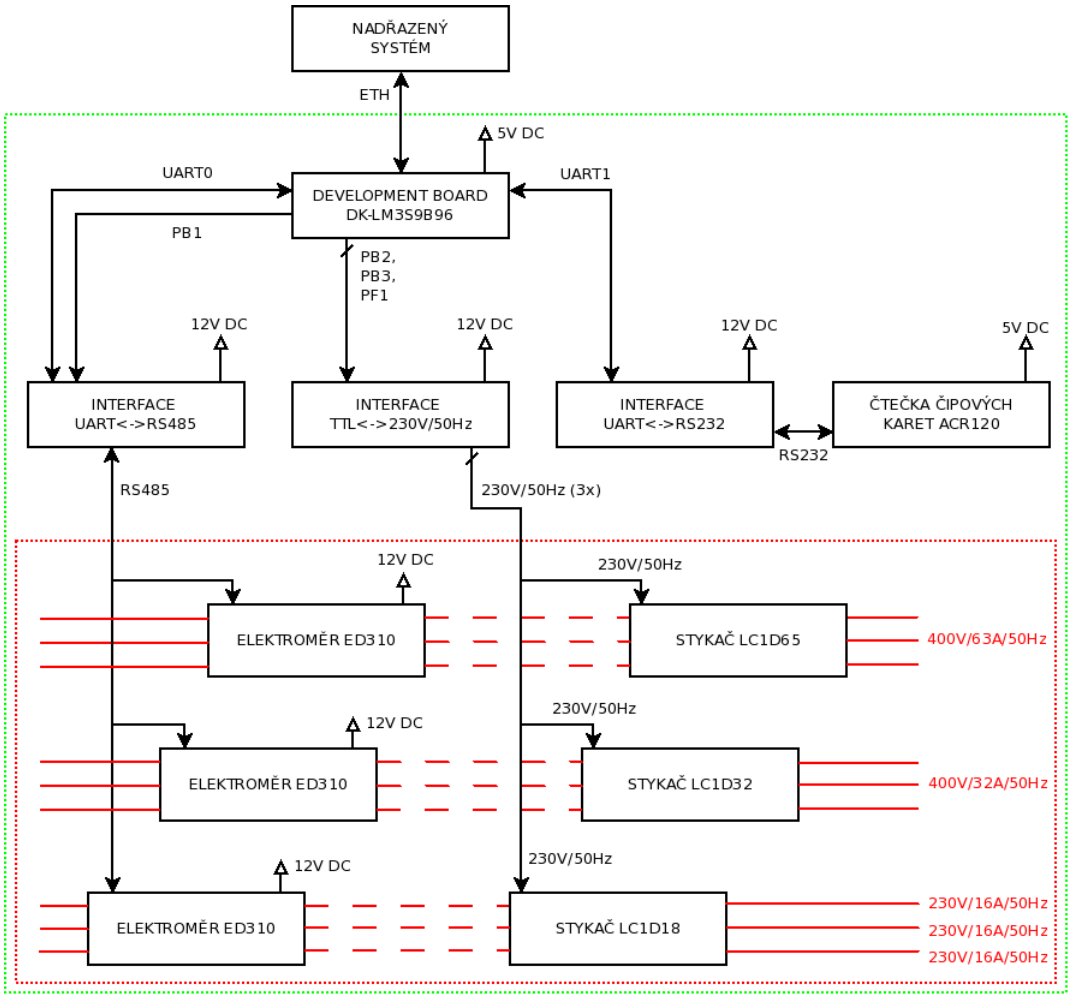
Obrázek 4: Rittal
(z [15])

3 PROTOTYP NABÍJECÍ STANICE

Náplní této kapitoly je přehled základních komponent prototypu bezobslužné nabíjecí stanice pro elektromobily realizovaného firmou ELNICO s.r.o.

3.1 Charakteristika prototypu

Prototyp automatické nabíjecí stanice pro elektromobily je elektrické zařízení, jehož činnost je řízena počítačem realizovaným vývojovým kitem DK-LM3S9B96. Systém umožňuje identifikaci uživatele čipovou kartou, nastavení požadovaného nabíjení a následný monitoring odebrané energie. K nabíjení lze využít třífázové zásuvky 400V/63A/50Hz a 400V/32A/50Hz, dále pak až tři zásuvky typu 230V/16A/50Hz (v rámci diplomové práce je však zapojena pouze jedna z 230 V zásuvek). Schéma systému znázorňuje obrázek 5.



Obrázek 5: Blokové schéma prototypu nabíjecí stanice

Zeleně orámovaná část zde tvoří ucelenou samostatně funkční jednotku - nabíjecí stojan. Bloky v červeném rámečku jsou pak částí silnoproudého systému, která je ovladatelná řídicím počítačem. Tvoří ji tři třífázové elektroměry ED310 a třífázové stykače LC1D65, LC1D32 a LC1D16.

K identifikaci zákazníka stanice používá čtečku čipových karet ACR120, celý systém je monitorován nadřazeným systémem - serverem.

Pro připojení výše uvedených komponent k řídicímu počítači stanice obsahuje:

- rozhraní UART1 <-> RS232 (pro připojení čtečky čipových karet),
- rozhraní UART0 <-> RS485 s řízením toku pinem PB1 mikrokontroléru LM3S9B96 (komunikace s elektroměry),
- rozhraní GPIO (piny PB2, PB3, PF1) <-> 230V/50Hz (ovládání stykačů).

3.2 Řídicí počítač

Při vývoji prototypu je jako řídicí počítač využíván vývojový kit DK-LM3S9B96, dostupný od společnosti Texas Instruments. Toto zařízení je vybaveno 32-bitovým mikrokontrolérem LM3S9B96, 3,5 palcovým barevným displejem a množstvím dalších vyvedených periférií připraveným k použití (viz. obrázek 6).

3.2.1 Mikrokontrolér LM3S9B96

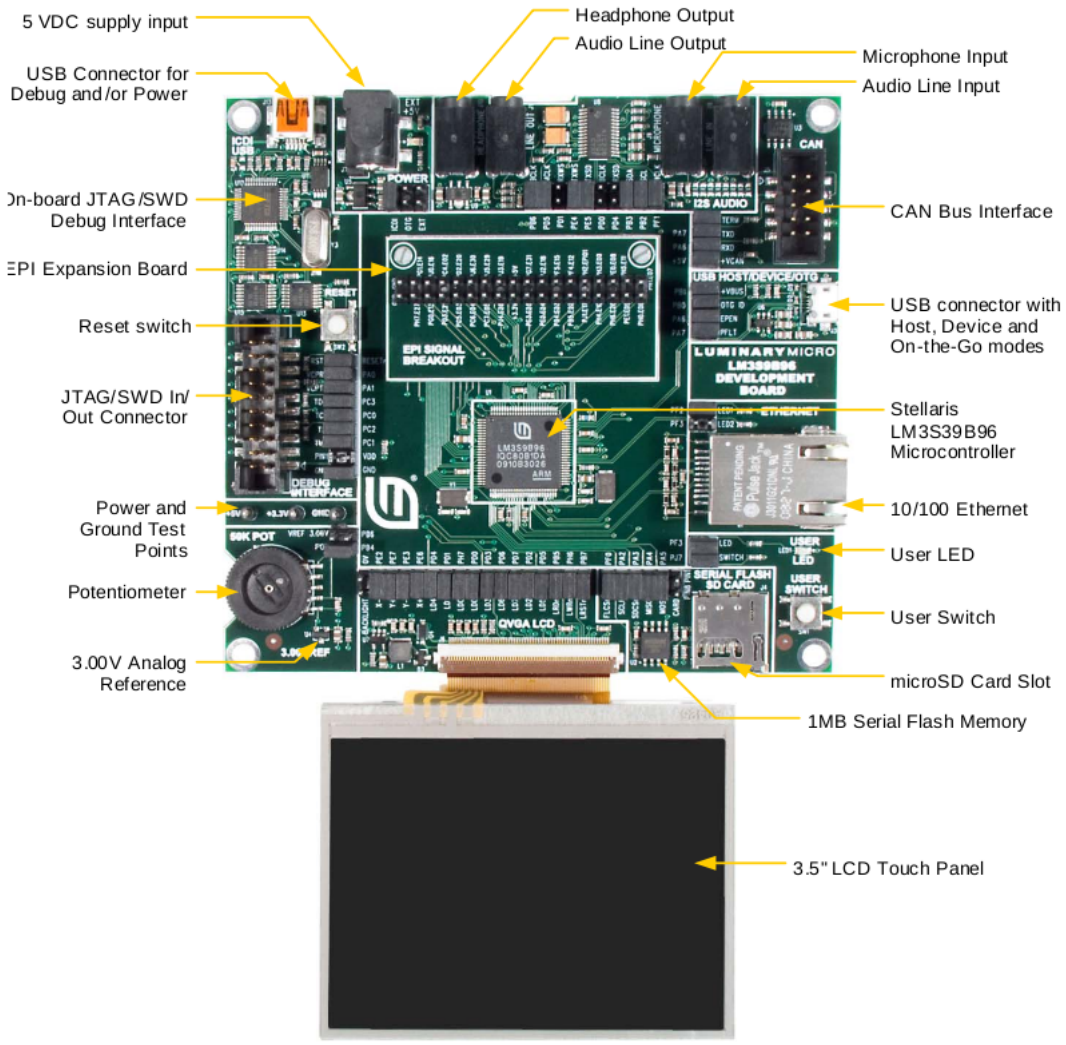
Tento 32-bitový mikrokontrolér je založen na jádře ARM Cortex-M3 a vyznačuje se harvardskou architekturou, charakteristickou oddělenými sběrnicemi pro kód a data. Napájen je 3,3 V napětím a taktován může být až na kmitočtech do 80 MHz. Teplotní rozmezí je udáváno -40...85 °C.

Na chipu je integrována 256 kB Flash, 96 kB SRAM a ROM paměť. Na té jsou předprogramovány boot loader, knihovna driverů periferií, jádro operačního systému SafeRTOS, kryptografické tabulky AES a funkce pro detekci chyb pomocí cyklického redundantního součtu (CRC).

Jádro mikrokontroléru je dále charakteristické instrukčním souborem složeným ze 16 i 32-bitových instrukcí (implementováno je např. násobení v jednom cyklu nebo hardwarové dělení). Důležitý je také 24-bit systémový časovač (SysTick), navržený k použití jako programovatelný RTOS timer.

3.2.2 Periférie kitu

Prototyp nabíjecí stanice využívá následující periferie vývojového kitu (z [10]):



Obrázek 6: Vývojový kit DK-LM3S9B96
(z [17])

Displej

Barevný LCD TFT displej firmy Kitronix je připojen paralelní 8-bitovou sběrnicí doplněnou řídicími signály. Zařízení poskytuje rozlišení 320x240 pixelů při 262 tisících barev a je vybaveno bílým LED podsvitem. Celkový rozměr je 3,5 palce. Obsažen je také rezistivní touch, připojený na 2 ADC a 2 GPIO piny mikrokontroléru.

Ethernetové rozhraní Pro komunikaci s nadřazeným systémem (serverem) je určeno konfigurovatelné ethernetové rozhraní. Na čipu jsou integrovány Media Access Controller (MAC) s nastavitelnou MAC adresou a fyzická vrstva (PHY). Podporovány jsou full a half-duplex operační módy při 10/100 Mbps.

EPI

EPI (External Peripheral Interface) je vysokorychlostní paralelní sběrnice o šířce 8/16/32 bitů. Využít lze k několika účelům, možné je:

- připojit paměť SDRAM o velikosti do 64 MB,
- připojit čip CPLD nebo FPGA,
- využít piny jako obecné vstupně výstupní rozhraní (GPIO).

Na vývojovém kitu DK-LM3S9B96 je na EPI sběrnici připojena 8 MB SDRAM paměť.

SD karta

Přes SPI rozhraní je připojen slot na microSD kartu. Součástí dodávaného kitu je karta o kapacitě 1 GB.

UART

Mikrokontrolér LM3S9B96 obsahuje celkem tři UART linky, z nichž jedna je přímo určena k sériové komunikaci i na vývojovém kitu. Další dvě linky lze však i bez HW úprav použít také, ačkoli jsou implicitně použity k jiným účelům.

GPIO

Čip obsahuje až 65 obecně využitelných vstupů/výstupů s napětím 3,3 V. Vstupy mohou vyvolávat přerušení na změnu úrovně, náběžnou či sestupnou hranu, zároveň jsou 5 V tolerantní. Proudy výstupů lze konfigurovat na 2 mA, 4 mA, 8 mA, či 18 mA.

3.3 Elektroměry

K měření odběru energie jsou určeny programovatelné třífázové elektroměry ED 310 dodávané společností ZPA Trutnov. Ty zaznamenávají pro jednotlivé fáze:

- odběr v kWh,
- okamžitou efektivní hodnotu proudu,
- činný výkon.

Navíc je uchováván celkový odběr v kWh na všech fázích dohromady, který je současně zobrazován na displeji elektroměru. Komunikace s ED310 je možná buď optorozhraním nebo galvanicky oddělenou sériovou linkou RS485.

Provozní teploty zařízení jsou udávány v rozsahu -25...50 °C. (vše z [22]).

Obrázek 7: Elektroměr ED 310
(z [22])

3.4 Stykače

Pro spínání napětí na jednotlivých zásuvkách jsou určeny stykače od firmy Schneider Electric. Použity jsou tyto typy:

- LC1D65 pro spínání třífázového napětí 400V/63A/50Hz,
- LC1D32 (třífázové 400V/32A/50Hz) a
- LC1D18 (jednofázové 230V/16A/50Hz).

Fotografie uvedených zařízení jsou k nahlédnutí na obrázku 8.

a) 

b) 

c) 

Obrázek 8: Stykače: a) LC1D65, b) LC1D32, c) LC1D18

3.5 Čtečka čipových karet

K identifikaci zákazníka je určena bezkontaktní čtečka čipových karet ACR 120 společnosti Advanced Card Systems Ltd. (viz. obrázek 9). Zařízení využívá RFID technologii a je schopno pracovat s kartami Mifare (Classic, DESFire) a kartami standardu ISO 14443 A i B. Maximální vzdálenost karty od čtečky je udávána 5 cm. Z komunikačních rozhraní lze volit USB či RS 232, zařízení je schopné funkce v rozsahu teplot 0...50 °C (vše z [2]).

Obrázek 9: Čtečka čipových karet ACR 120
(z [2])

3.6 Nadřazený systém

Jednotlivé nabíjecí stanice jsou určeny k zapojení v síti s centrálním bodem - serverem. K němu jsou připojeny ethernetovou linkou a mohou s ním komunikovat stylem klient-server. Systém obsahuje především databázi zákazníků specifikovaných minimálně:

- identifikačním číslem čipové karty,
- stavem předplaceného kreditu.

Dalším úkolem serveru je pak monitoring stavu podřízených stanic.

4 POROVNÁNÍ S KONKURENČNÍM PRODUKTEM

V této kapitole je porovnána bezobslužná nabíjecí stanice pro elektromobily realizovaná firmou ELNICO s.r.o. s obdobným systémem společnosti Rittal. Právě tato firma vyhrála v březnu 2010 výběrové řízení ČEZu na rozmístění prvních nabíjecích stanic v ČR.

	ELNICO	**Rittal**
Počet odběrných míst	5	2
Konfigurace odběrných míst	400V/63A/50Hz, 400V/32A/50Hz, 230V/16A/50Hz	400V/50Hz, 230V/50Hz, konfigurovatelně 16-63A
Identifikace zákazníka	Čipové karty Mifare (Classic, DESFire) a ISO 14443 A i B	Především čipové karty Mifare Classic
Vzdálený monitoring zákazníkem	Nemá	Modul GSM/GPRS
Vzdálený monitoring provozovatelem	Nadřazený systém - server	Není uvedeno
Možnosti umístění	Spíše Indoor	Indoor i outdoor

Tabulka 1: Porovnání nabíjecích stanic firem ELNICO a Rittal

Z tabulky 1 jsou patrné základní rozdíly mezi stanicemi. Srovnatelné jsou oba systémy z hlediska typů obsažených zásuvek a způsobu identifikace zákazníka. Stanice firmy ELNICO obsahuje vyšší počet odběrných míst - pět oproti dvěma u konkurenčního modelu. Mezi výhody systému firmy Rittal patří naopak vzdálený monitoring zákazníkem díky GSM/GPRS modulu a možnost outdoor použití.

5 OPERAČNÍ SYSTÉM FREERTOS

V této kapitole je uveden základní popis OS FreeRTOS, který je použit při tvorbě aplikace.

5.1 Charakteristika systému

Jak název celé kapitoly napovídá, FreeRTOS je operační systém reálného času. Určen je zejména pro embedded zařízení a mezi jeho hlavní vlastnosti patří především malá velikost, jednoduchost a široká portabilita. Podporovány jsou architektury ARM7, ARM Cortex-M3, Atmel AVR, PIC32 a mnoho dalších. Napsán je v jazyce C a distribuován je pod licencí GPL s výjimkou, díky níž je možné vytvořený kód aplikace uzavřít (informace dostupné z [20]).

V dalších podkapitolách budou popisovány:

- úlohy a jejich plánování,
- fronty zpráv, které je možné využít k synchronizaci mezi úlohami,
- řešení exkluzivního přístupu ke zdrojům,
- přerušení.

5.2 Úlohy a jejich plánování

Základní vlastností každého operačního systému je rozčlenění řešeného problému do více samostatných úloh. Každá úloha je v OS FreeRTOS charakterizována:

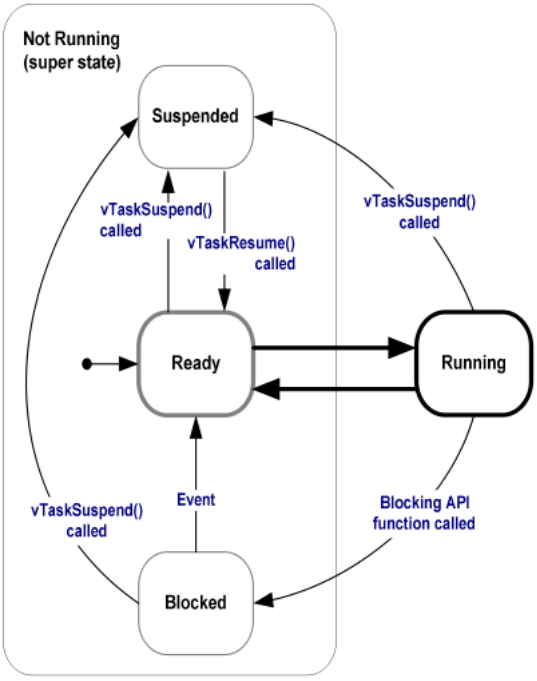
- Názvem,
- exekučním kódem,
- velikostí stacku,
- prioritou,
- vstupními parametry,
- referencí (handlem).

Úlohy se mohou nacházet v jednom z několika stavů, jak naznačuje obrázek 10. Implicitní je stav Ready, do kterého úloha vstoupí po svém vytvoření. Pokud je

poté naplánovana plánovačem systému k běhu, změní svůj stav na Running a vykonává svůj exekuční kód.

Musí-li úloha ve stavu Running čekat na přístup k nějakému zdroji, je přepnuta do stavu Blocked. Je-li později zdroj uvolněn, změní úloha svůj stav na Ready a čeká, až bude opět naplánována, aby k němu mohla přistoupit.

Posledním stavem, ve kterém se úloha může nacházet, je Suspended. Přesun do a z tohoto stavu je možné pomocí API funkcí.



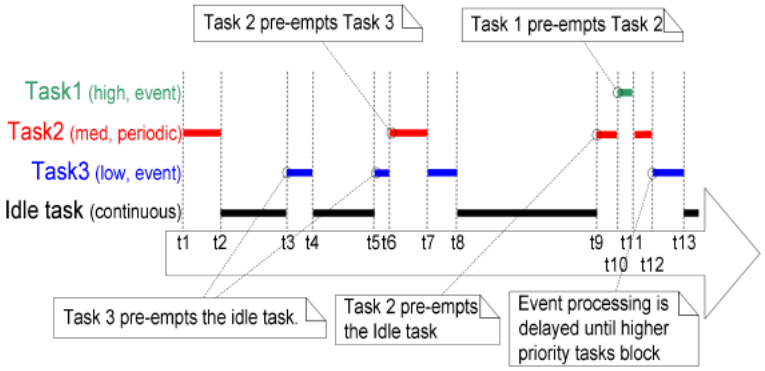
Obrázek 10: Stavový diagram úlohy
(z [5])

V jednom okamžiku může být ve stavu Running právě jedna úloha. K určení toho, která to bude, obsahuje OS FreeRTOS plánovač systému - preemptivní s fixními prioritami úloh (jádro systému nemůže měnit jejich priority). Preemptivita pak spočívá v tom, že plánovač kvůli každé úloze, která má vysokou prioritu a nachází se ve stavu Ready, přeruší běh jakékoli úlohy s nižší prioritou.

Priority úloh mohou nabývat hodnot od 0 (nejnižší) do konstanty configMAX_PRIORITIES, kterou je nutné specifikovat v hlavičkovém souboru před kompilací. OS FreeRTOS jinak sám o sobě maximální možný počet priorit neomezuje.

Možná aktivita úloh OS FreeRTOS v čase je znázorněna na obrázku 11. Každá aplikace obsahuje tzv. Idle task, tedy úlohu s nejnižší prioritou, která je přerušena jakoukoli naplánovanou úlohou. V příkladu na obrázku jsou další 3 úlohy, z nichž

každá má jinou prioritu. Úlohy Task 1 a Task 3 jsou řízené událostmi, úloha Task 2 je pak aktivovaná periodicky. Jak je z obrázku patrné, úloha s vyšší prioritou vždy přeruší úlohu s prioritou nižší a sama se dostane do stavu Running.



Obrázek 11: Aktivita úloh v čase
(z [5])

5.3 Fronty

V multitaskingovém systému nastává nutnost komunikace mezi jednotlivými úlohami. Jedním z prostředků, které toto umožňují, jsou v OS FreeRTOS tzv. fronty (queues).

Každá fronta je v podstatě jednoduchý zásobník obvykle typu FIFO, který slouží k přenosu dat mezi dvěma a více úlohami (nejčastěji jeden nebo více zapisovatelů a pouze jeden čtenář). Fronty jsou specifikovány typem dat (např. integer, double nebo definovaná struktura) a délkou (kolik položek daného typu se do fronty vejde).

Při pokusu o čtení z fronty může být úloha blokována, a to ze dvou důvodů:

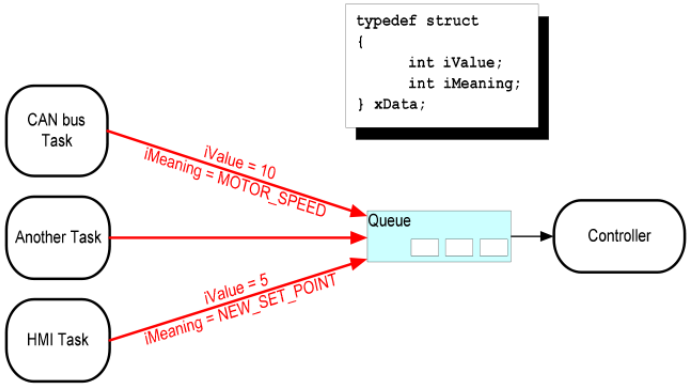
1. Ve frontě nejsou žádná data - pak se čeká na její naplnění;
2. z fronty se snaží číst najednou více úloh - v tom případě přístup získá pouze úloha s nejvyšší prioritou, po ní až ostatní.

Obdobná situace nastává při pokusu o zápis do fronty. K blokování dochází, pokud:

1. Je fronta plná - čeká se na uvolnění místa v ní;
2. do fronty se snaží zapisovat více úloh najednou - přístup získá pouze úloha s nejvyšší prioritou, až poté další.

Jak při čtení, tak při zápisu do fronty je specifikován „blokovací" čas - tzn. maximální doba (lze specifikovat i nekončený čas), po kterou úloha zůstává ve stavu Blocked. Nejpozději po jejím uplynutí změní stav zpět na Ready, tím je však odeslaná zpráva nenávratně ztracena.

Možné konkrétní použití fronty si ukážeme na obrázku 12. Úloha *Controller* zpracovává vstupy čtením dat z fronty, která obsahuje tři položky typu xData. K zápisu jsou oprávněny tři úlohy, z nichž každá může mít jinou prioritu. Úloha *CAN bus Task* ukládá do fronty aktuální rychlost motoru, úloha *HMI Task* pak nastavenou pozici. *Controller* pak typ dat pozná podle příznaku iMeaning.



Obrázek 12: Příklad využití fronty
(z [5])

5.4 Exkluzivní přístup ke zdrojům

Ve všech multitaskingových systémech existuje riziko vzniku problému při nedodržení exkluzivního přístupu ke zdrojům. Stejně tak tomu je i v případě OS FreeRTOS.

Problém vysvětlíme na příkladu dvou úloh, vypisujících text na displej (první tiskne text „Task number 1", druhá „Task number 2"). První úloha je ve stavu Running a začne vypisovat. V okamžiku, kdy je vytištěno „Task nu", je však plánovačem aktivována úloha druhá, která má vyšší prioritu. Ta vypíše svůj text „Task number 2" a je přepnuta zpět do stavu Ready. Teprve nyní první úloha vypíše zbytek své zprávy. Výsledný text na displeji je pak „Task nuTask number 2mber 1".

Podobná nepříjemnost může vznikat např. při neatomickém přístupu ke globálním proměnným nebo při volání nereentrantních funkcí. Všechny případy se však shodují v tom, že obsahují určitý exekuční kód (kritická sekce), ke kterému je potřeba přistupovat exkluzivně.

V OS FreeRTOS existuje několik mechanismů, kterými lze exkluzivní přístup zajistit:

- Suspendování plánovače,
- binární semafor,
- mutex,
- tzv. gatekeeper úloha.

5.4.1 Suspendování plánovače

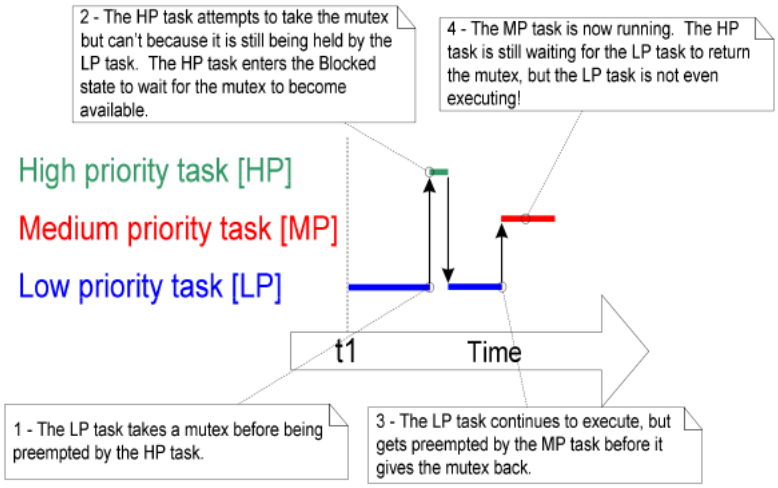
Tento nejjednodušší mechanismus spočívá v pozastavení (suspendování, zamknutí) funkce plánovače před vstupem do kritické sekce a v povolení (odemknutí) jeho funkce po výstupu z ní.

Kritická sekce je pak chráněna před přístupem z jiných úloh, ošetřen však není přístup z obsluh přerušení.

5.4.2 Binární semafor

Řízení přístupu do kritické sekce lze provádět také pomocí semaforu. Ten si lze představit jako token, který je asociovaný se sdíleným zdrojem. Pokud chce úloha přistoupit ke zdroji, musí nejdříve požádat o token (semafor). Nedostane-li ho, je přepnuta do stavu Blocked. Dostane-li ho, může do kritické sekce vstoupit a provést potřebné operace. Poté musí token (semafor) zase uvolnit, aby ke zdroji mohly přistupovat ostatní úlohy.

Při použití binárních semaforů může docházet k tzv. inverzi priorit nebo dokonce k deadlocku. Inverzi priorit lze popsat podle obrázku 13: Úloha s nízkou prioritou (LP) přistupuje ke zdroji a díky semaforu tedy blokuje úlohu s vysokou prioritou (HP). To je v pořádku, problém však nastává, když se do stavu Running přepne třetí úloha s prostřední prioritou (MP), která nevyžaduje přístup ke zdroji. Vysokoprioritní úloha (HP) pak musí čekat, než MP úloha dokončí svou činnost.



Obrázek 13: Inverze priorit
(z [5])

Daleko závažnějším problémem je deadlock. Ten může vzniknout např. v situaci, kdy každá ze dvou úloh potřebuje získat exkluzivní přístup ke dvěma zdrojům A a B.

Postup vzniku deadlocku:

1. První úloha získá přístup ke zdroji A.
2. Poté je naplánována druhá úloha, která získá přístup ke zdroji B a snaží se získat přístup také ke zdroji A. To se jí nepodaří (ke zdroji přístupuje první úloha), je tedy přepnuta do stavu Blocked.
3. Úloha A je opět naplánována a nyní se snaží získat přístup ke zdroji B, který není dostupný - také přepnutí do stavu Blocked.
4. Deadlock - ani jedna úloha nemůže pokračovat.

Při použití semaforů je důležité navrhovat systém takovým způsobem, aby k deadlocku nemohlo dojít.

5.4.3 Mutex

Mutex je v podstatě binární semafor. V operačním systému FreeRTOS se však od klasického semaforu liší tím, že využívá tzv. inheritanci (dědění) priorit. Tím je vyřešen problém inverze priorit, který se vyskytuje u binárních semaforů.

Jak dědění priorit funguje, lze popsat na jednoduchém příkladu: Úloha A s nízkou prioritou získá mutex. Poté o něj žádá jiná úloha B, která má vysokou pri-

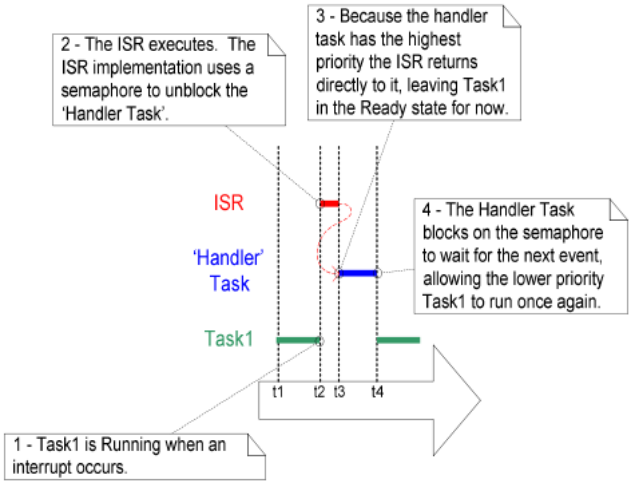
oritu. Aby nemohlo dojít k problému inverze priorit, zdědí proto úloha A prioritu úlohy B.

5.4.4 Gatekeeper úloha

Úloha typu Gatekeeper[1] poskytuje čistý způsob implementace exkluzivního přístupu ke zdroji, při němž nemůže docházet k problémům jako je inverze priorit nebo deadlock (z [5]). Princip spočívá v tom, že ke zdroji, který již není globální, je přistupováno pouze z gatekeeper úlohy. Využívána je fronta s několika zapisovateli a jedním čtenářem (gatekeeper).

5.5 Přerušení

Zejména pro korektní zacházení s periferiemi procesoru musí operační systém obsahovat nástroje pro obsluhu přerušení. Pro každý typ přerušení, které chceme nějakým způsobem obsluhovat, je nutné definovat obslužnou rutinu. V OS FreeRTOS je však doporučováno, aby tato rutina provedla pouze nejnutnější operace a aby vlastní obsluha byla předána speciální úloze. Tak to ostatně popisuje i obrázek 14.



Obrázek 14: Aktivita úloh v čase při přerušení (z [5])

Při přerušení procesor okamžitě skočí do jeho ISR (Interrupt Service Routine). ISR může klidně provést celou obsluhu, to však není úplně korektní, protože v systému mohou běžet některé úlohy s vysokou prioritou, které v tu chvíli potřebují být aktivní (Running). Proto je v systému ještě speciální úloha, určená pro

[1] Gatekeeper - vrátný

obsluhu daného přerušení a implicitně blokovaná binárním semaforem. Při přerušení je ale v ISR tento semafor odblokován a již zmíněná úloha se dostane do stavu Ready. Pokud v té chvíli v systému neexistují jiné úlohy s vyšší prioritou, pak je naplánována a provede obsluhu přerušení.

Způsob práce s binárním semaforem se v tomto případě od použití v kritických sekcích (viz. kapitola 5.4.2) liší. Zatímco v kritických sekcích lze bin. semafor chápat jako token, který střídavě vlastní jedna z několika úloh, pro synchronizaci při přerušení je v [5] popisován jako „fronta s jedním prvkem". Do té je možné „zapisovat" (uvolnění bin. semaforu) či z ní „číst" (získání). Rozdíl je v tom, že po získání semaforu není nutné ho opět uvolnit - „fronta" je totiž prázdná a je možné do ní opět „zapisovat". ISR tedy semafor při vyvolání pouze uvolňuje a úloha určená k obsluze přerušení pouze zjišťuje jeho dostupnost.

6 ŘÍDICÍ APLIKACE - STRUKTURA

Praktickou náplní diplomové práce je tvorba softwarového vybavení pro řídicí počítač prototypu bezobslužné nabíjecí stanice určeného k nabíjení elektromobilů. Tato kapitola popisuje základní vlastnosti vytvořeného programu a uvádí v přehledu používané úlohy a sdílené zdroje.

6.1 Charakteristika vytvořeného SW

Ze zadání firmy ELNICO s.r.o. plyne několik základních požadavků na vytvářený program:

- možnost výběru odběrného místa,
- zajištění identifikace uživatele využitím čtečky čipových karet ACR120,
- možnost nastavení požadovaného odběru,
- monitoring průběhu nabíjení pomocí elektroměrů ED 310 a
- ovládání (start, stop) nabíjecího procesu prostřednictvím stykačů LC1D65, LC1D32 a LC1D18.

V rámci diplomové práce tedy není realizována komunikace s nadřízeným systémem - serverem.

Pro psaní kódu a debugování aplikace je použito vývojové prostředí Red Suite 2.0.16 společnosti Code Red, přesněji jeho 90-denní evaluation verze. Program je napsán v programovacím jazyku C a využívá:

- knihovnu Driverlib,
- firmwarový balíček pro vývojový kit DK-LM3S9B96,
- grafickou knihovnu Grlib,
- OS FreeRTOS.

Nad těmito knihovnami a operačním systémem je vytvořen program s pracovním názvem elcharger[2]. Obsahuje moduly, které pracují s:

1. reálným (systémovým) časem,

[2]elcharger: el - Elnico, charger - nabíjecí systém

2. SD kartou (na kterou je logováno),
3. grafickými objekty (určenými k zobrazení na displeji),
4. displejem (jako rozhraním pro ovládání aplikace uživatelem),
5. čtečkou čipových karet (sloužící k identifikaci uživatele),
6. elektroměry (kterými je měřen odběr na jednotlivých nabíjecích místech),
7. stykači (určenými ke spínání napětí na těchto nabíjecích místech),
8. kontrolou průběhu (monitoringem) nabíjení.

Všechny tyto moduly jsou podrobněji popsány v kapitole 7.

6.2 Úlohy

Celý program je členěn do devíti úloh, z nichž každé je přiřazen název, priorita a oblast paměti určité velikosti ve stacku. Jejich přehled se základními vlastnostmi a stručnými popisky je uveden v tabulce 2.

Název	Priorita	Stack	Řízení	Popis
RT	9	50 B	na událost	Na přerušení od Timeru inkrementuje systémový čas
Cardrdr	8	50 B	na událost	Zpracovává data přijatá z UARTu čtečky karet
Elmeter	7	150 B	na událost	Zpracovává data přijatá z UARTu elektroměrů
Charging	4	100 B	periodické	Zasílá elektroměrům požadavky na informace o odběru energie, monitoruje a ovládá nabíjení
LogGatekpr	5	400 B	na událost	Čte z fronty chybových zpráv a zapisuje do souboru na SD kartě
DispGatekpr	6	200 B	na událost	Čte z fronty displeje a jako jediná úloha na displej vypisuje
DispUpdater	3	100 B	periodické	Každých 500 ms zjišťuje, zda je potřeba překreslit čas nebo objekt (v případě ukončení nabíjení) na displeji
DispCrdrdr	1	150 B	periodické	Animace ikony čtečky karet, dotazování čtečky karet na ID přiložené karty
DispControl	2	100B	periodické	Způsobuje aktualizaci informací o nabíjení na displeji

Tabulka 2: Seznam úloh

Priority jsou nastaveny takovým způsobem, aby úlohy s delším exekučním kódem na jednotku času měli přiřazenu nižší hodnotu. Zároveň žádné dvě úlohy nemají stejnou prioritu, tzn. vždy je jasné, jaká úloha bude plánovačem naplánována. Trochu se vymykají úlohy DispUpdater, DispCrdrdr a DispControl, kterým jsou nastaveny nižší priority spíše z důvodu menší důležitosti operací, které v nich jsou prováděny.

Alokovaný stack odpovídá v podstatě paměti, kterou úloha využívá. Jsou do něj ukládány lokální proměnné úlohy i proměnné z ní volaných funkcí. Úlohy programu elcharger potřebují paměť řádově ve stovkách bytů. Nejnáročnější jsou ty, které provádí operace s řetězci.

V tabulce 2 jsou úlohy dále specifikovány způsobem, jakým může být jejich stav změněn do READY. Možnosti jsou obecně celkem dvě:

1. po výskytu nějaké události (event-driven),
2. periodicky (či obecně v závislosti na čase, tedy time-driven).

6.3 Sdílené zdroje

Úlohy jsou samostatné funkční celky, které pracují s určitými zdroji (daty, proměnnými), většinou lokálními. Problém nastává, chceme-li přistupovat z více úloh ke zdrojům globálním. V takovém případě je nutné zajistit exkluzivní přístup. OS FreeRTOS poskytuje k tomuto účelu dva základní prostředky (podrobněji v kapitolách 5.3 a 5.4) - využít lze:

- fronty zpráv,
- globální proměnné chráněné v kritické sekci semaforem.

Přehled front použitých v programu elcharger je uveden v tabulce 3. Každá fronta je charakterizována svým jménem, délkou (maximálním počtem položek), vlastníkem (tedy úlohou, která jediná je oprávněná z ní číst) a typem obsahu. Do fronty *g_xLogQueue* mohou zapisovat všechny běžící úlohy, do fronty *g_xDisplayQueue* pouze DispUpdater, DispCardrdr a DispControl.

Identifikátor fronty	Délka	Vlastník	Typ obsahu
g_xLogQueue	5	LogGatekpr	Logovací zprávy
g_xDisplayQueue	5	DispGatekpr	Zprávy popisující „co a jak vykreslit na displej”

Tabulka 3: Seznam použitých front

Stejně tak jsou použity globální proměnné, k nimž je přistupováno vždy ze dvou různých úloh. V tomto případě je zajištěn exkluzivní přístup použitím semaforu. Zdroje jsou shrnuty v tabulce 4.

Sdílená proměnná	Semafor	Popis obsahu proměnné
g_uiCardrdrId	*g_xCardrdrSemph*	Číslo přiložené čipové karty (zápis úloha Cardrdr, čtení DispCardrdr)
g_Elmeter[]	*g_xElmeterSemph*	Informace přijaté z elektroměrů (zápis Elmeter, čtení Charging)
g_Charging[]	*g_xChargingSemph*	Informace o nabíjení (zápis Charging, čtení DispControl)
g_Rt	*g_xRtSemph*	Systémový čas (zápis Rt, čtení DispUpdater)

Tabulka 4: Seznam globálních proměnných synchronizovaných mezi úlohami

6.4 Obsluhy přerušení

Jak již bylo teoreticky uvedeno v kapitole 5.5, v OS FreeRTOS je doporučováno provádět obsluhu vyvolaného přerušení až v úloze, která má přiřazenou určitou prioritu. To platí především pro ty typy přerušení, jejichž obsluha vyžaduje provedení většího množství programového kódu.

Přehled přerušení obsluhovaných částečně obsluhovací rutinou (ISR) a částečně úlohou je uveden v tabulce 5. Přerušení, která jsou obsluhována přímo, shrnuje tabulka 6.

Přerušení	Úloha	Semafor	Popis obsluhy
TIMER0	RT	*g_xRtIntSemph*	V ISR je uvolněním semaforu odblokována úloha RT, v níž dochází k aktualizaci sys. času
UART0	Elmeter	*g_xElmeterIntSemph*	Po přijetí celého telegramu z elektroměru odblokuje ISR úlohu Elmeter, která příchozí data zpracuje
UART1	Cardrdr	*g_xCardrdrIntSemph*	Po příjetí celé zprávy způsobí ISR odblokování úlohy Cardrdr, která data zpracuje a získá ID karty

Tabulka 5: Seznam nepřímo obsluhovaných přerušení

Zdroj přerušení	Popis obsluhy
TIMER1	Přerušení o frekvenci 100 Hz slouží jako zdroj pulzů pro souborový systém FAT na SD kartě
Touch screen	Z ISR je přímo zaslána zpráva se souřadnicemi bodu dotyku displeje do fronty úlohy DispGatekpr

Tabulka 6: Seznam přímo obsluhovaných přerušení

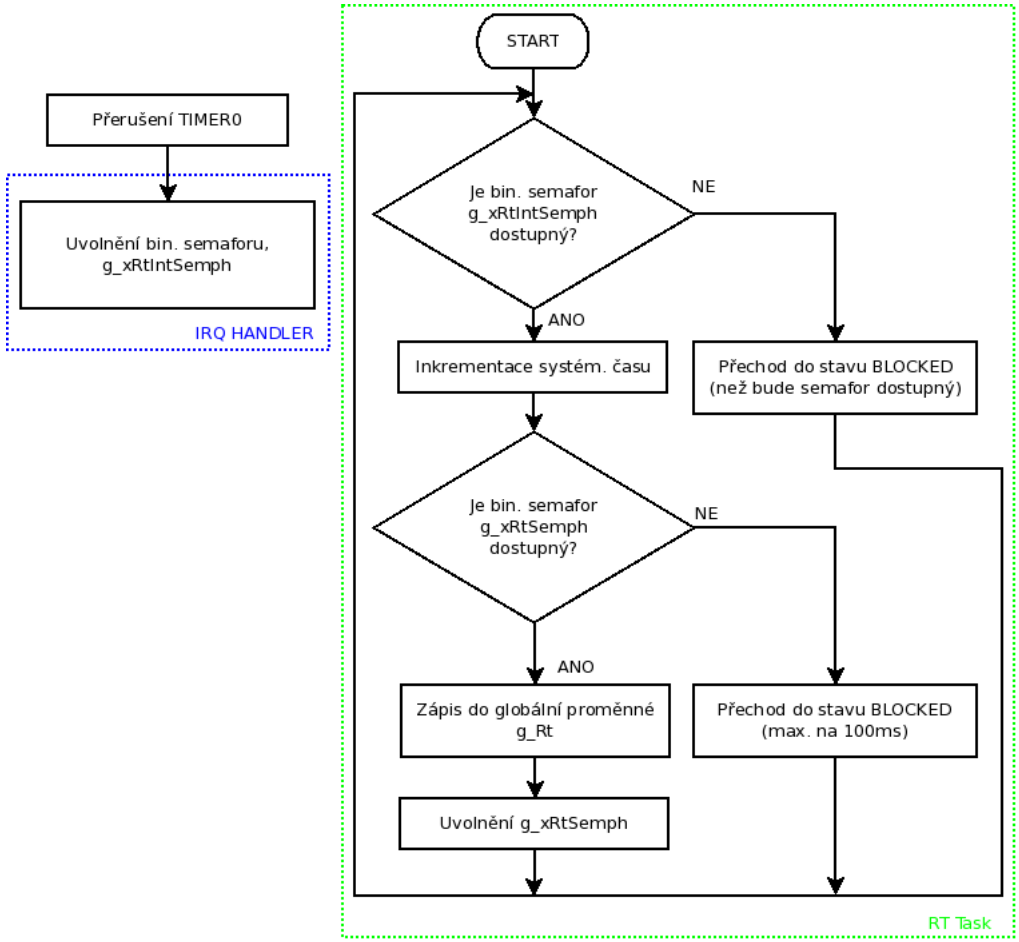
7 ŘÍDICÍ APLIKACE - MODULY

Při práci na projektu bylo snahou vytvořit program co nejvíce modulární. Jednotlivých funkčních celků je obsaženo celkem osm a jejich výčet je uveden v kapitole 6.1. Podrobný popis těchto modulů je náplní této kapitoly.

7.1 Uchovávání reálného času

Nejmenším a zároveň nejjednodušším samostatně funkčním článkem aplikace je uchovávání reálného času reprezentované především úlohou RT a funkcemi, jejichž voláním z ostatních úloh lze aktuální čas zjistit. Časové údaje jsou potřebné zejména při zápisu do logu, zobrazovány jsou pro informaci uživatele i na displeji.

Úloha RT reaguje na událost (periodicky vyvolávané přerušení časovače Timer 0), má vysokou prioritu 9 a ke svému běhu potřebuje pouze 50 B alokované paměti ve stacku. Její funkce je popsána vývojovým diagramem na obrázku 15.



Obrázek 15: Vývojový diagram úlohy RT

V obsluze přerušení Timeru 0, které je generováno každou 1 s, je uvolněn binární semafor *g_xRtIntSemph*. Tím je odblokována úloha RT, ve které okamžitě dochází k inkrementaci systémového času (pozn.: uchováván je ve struktuře, tzn. sekundy, minuty a hodiny zvlášť).

Pro umožnění přístupu k tomuto času ostatním úlohám je v kritické sekci chráněné binárním semaforem *g_xRtSemph* aktuální hodnota zapsána do globální proměnné. Zde je vhodné poukázat na odlišnost práce s oběma semafory. Zatímco na dostupnost semaforu *g_xRtIntSemph* úloha může čekat klidně nekonečný čas (aby se dostala ze stavu BLOCKED do RUNNING), u *g_xRtSemph* musí být maximální doba čekání stanovena na menší než 1 s. Kdyby tomu tak nebylo, mohlo by docházet ke ztrátě informace, systémový čas by mohl být „zpomalován".

K úloze RT je nakonec také vhodné zmínit, že důsledkem absence obvodu typu time keeper na vývojovém kitu DK-LM3S9B96 je systémový čas po každém spuštění aplikace počítán od nuly.

7.2 Logování na SD kartu

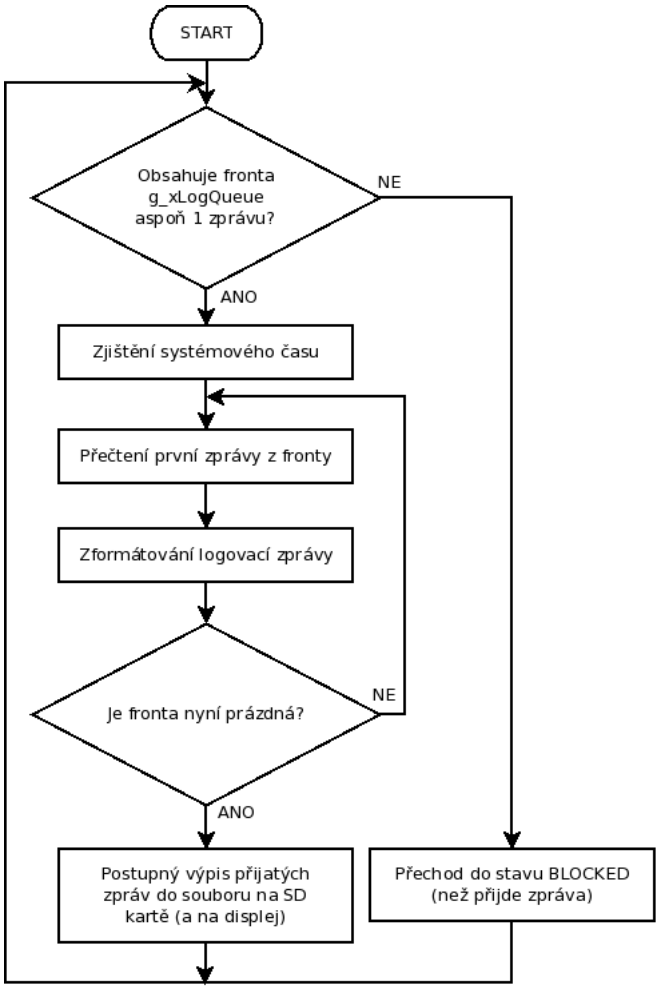
Důležitou funkcí každé aplikace musí být možnost ukládání informačních a chybových zpráv do logu. Ten je pro program elcharger umístěn v souborovém systému na SD kartě a zápis do něj je umožněn přes úlohu LogGatekpr. Tato úloha reaguje na událost (příchod zprávy), má středně vysokou prioritu 5 a využívá největší stack paměť ze všech úloh - 400 B. Důvodem vysoké spotřeby paměti je práce s řetězci.

Úloha LogGatekpr vlastní systémový zdroj - frontu *g_xLogQueue* (typ FIFO). Zapisovat do ní mohou všechny běžící úlohy a v jednom okamžiku v ní může být obsaženo až 5 logovacích zpráv. A právě aktuální počet zpráv ve frontě má přímý vliv na stav úlohy. Ta se totiž v okamžiku, kdy je naplánována, snaží přečíst první zprávu z fronty. Pokud je fronta prázdná, pak je stav úlohy změněn na BLOCKED. V opačném případě, tedy když ve frontě je přítomna alespoň jedna zpráva, dojde k jejímu uložení a zformátování dohromady se systémovým časem do jednoho řetězce.

Takto je z fronty čteno do té doby, dokud v ní je alespoň jedna nezpracovaná zpráva. Poté jsou všechny (zformátované se systémovým časem) zapsány do souboru na SD kartě a zároveň (kvůli ladicím účelům) na displej. Příklad začátku logovacího souboru:

```
00:00:01 INFO: Log file successfully created
```

Mechanismus úlohy je znázorněn vývojovým diagramem na obrázku 16.



Obrázek 16: Vývojový diagram úlohy LogGatekpr

7.3 Sada grafických objektů

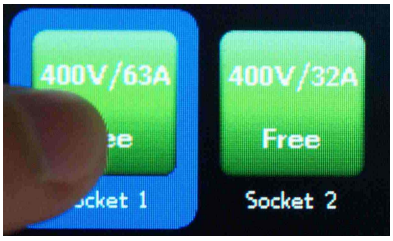
Společnost Texas Instruments poskytuje k distribuovaným vývojovým kitům mj. grafickou knihovnu grlib. Ta obsahuje fonty, objekty (tlačítka, textová pole, ...) a funkce pro vykreslování na displej. Zejména způsob práce s těmito objekty se však pro použití s OS FreeRTOS ukázal jako nevyhovující, proto byl vytvořen vlastní modul, který původní nahrazuje.

7.3.1 Struktura Widget

Ústředním prvkem realizovaného modulu je „widget" - grafický objekt, který může představovat tlačítko či textové pole. Struktura widgetu má několik základních položek:

- popisek (titulek),
- ukazatel na přidružený základní obrázek,
- flag, zda má prvek definovanou odezvu pro „kliknutí" na dotykovém displeji,
- flag, zda má být prvek zobrazen.

Vzhled widgetu lze demonstrovat obrázkem 17. Zobrazena jsou zde dvě tlačítka, z nichž jedno je právě aktivované dotykem prstu. Oba objekty obsahují stejný obrázek a tři popisky.



Obrázek 17: Widget

7.3.2 Struktura Screen

Screen, tedy obrazovka, je struktura obsahující určité pozadí a konečnou množinu widgetů. Tento objekt byl definován zejména z toho důvodu, aby bylo možné pracovat najednou se skupinou grafických objektů. Velmi důležitou funkcí je *WidgetMessageProcess ( )*, která je volána při přerušení od dotykového displeje. Funkce v závislosti na souřadnicích „kliknutí" a typu zprávy („PTR_DOWN", „PTR_UP") určí, zdá má být pro nějaký widget volán handler.

Možný vzhled obrazovky tvořené třemi widgety znázorňuje obrázek 18.

Obrázek 18: Screen

7.3.3 Obrázky

Jak je již patrné z kapitoly 7.3.1, widgety jsou tvořeny obrázky, které musí být někde uloženy. Jako jejich permanentní úložiště je použita složka graphics na SD kartě, kde jsou jednotlivé prvky reprezentovány binárními soubory. Přístup na SD kartu je dosti pomalý, proto jsou všechny používané obrázky po startu programu překopírovány do globálních proměnných v externí paměti SDRAM s kapacitou 8 MB. Řídicí aplikace využívá obrázky o celkové velikosti přibližně 270 kB.

7.4 Vykreslování na displej

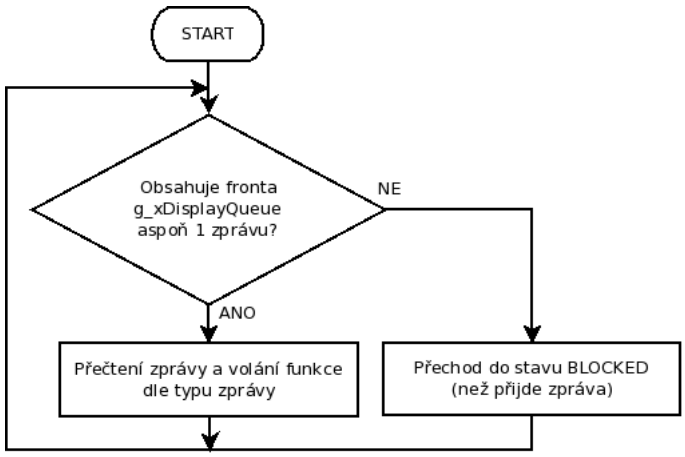
Ucelenou částí programu, která grafické objekty využívá, je modul pro vykreslování na displej. Hlavní činnosti jsou zde prováděny úlohou DispGatekpr, která jako jediná na displej přistupuje přímo. Použita je však ještě další skupina úloh, které na displej sice přímo nevykreslují, ale zobrazené objekty mohou ovlivňovat. Jsou to DispUpdater, DispCardrdr a DispControl.

7.4.1 Přímý přístup na displej

Jak již bylo uvedeno, překreslování displeje je řešeno pouze úlohou DispGatekpr. Ta plní roli „vrátného": vlastní frontu *g_xDisplayQueue* o maximálním počtu 5-ti položek. Základní parametry úlohy jsou: reakce na událost (přítomnost zpávy ve frontě), poměrně vysoká priorita 6 a paměť ve stacku o velikosti 200 B.

Požadavky na vykreslení určitého obsahu na displej se dosti liší. Pravidelně potřebujeme vykreslit systémový čas (což je řetězec dané délky), stejně tak je ale nutné např. změnit aktuální obrazovku či překreslit tlačítko při jeho stisknutí (oba tyto případy již není vhodné specifikovat řetězcem). Proto každá položka fronty obsahuje kromě parametrů zprávy (např. tedy již zmíněný řetězec nebo ukazatel na obrazovku, kterou chceme vykreslit) ještě identifikátor (tedy informace „jak s daným obsahem pracovat"). Tím je zajištěna možnost zpracování v podstatě libovolného typu zprávy.

Mechanismus úlohy lze znázornit vývojovým diagramem na obrázku 19.



Obrázek 19: Vývojový diagram úlohy DispGatekpr

Přehled hodnot identifikátorů zpráv a jim příslušných funkcí je pak zobrazen v tabulce 7.

Ident.	Volaná funkce	Popis funkce
1	DisplayGatekeeperTouchFcn	V závislosti na souřadnicích X a Y, ve kterých byl stlačen displej, je např. volána obsluha stisku tlačítka
2	DisplayGatekeeperScreenFcn	Změna aktuální obrazovky
3	DisplayGatekeeperWidgetFcn	Překreslení widgetu (tlačítka, ikony)
4	DisplayGatekeeperRtFcn	Překreslení systémového času
5	DisplayGatekeeperLogFcn	Výpis logovací zprávy na displej
6	DisplayGatekeeperCalcFcn	V závislosti na parametrech zapisuje do textového pole pro nastavení nabíjení
7	DisplayGatekeeperCardidFcn	V závislosti na ID karty přiložené ke čtečce povolí/nepovolí uživateli přístup k nabíjecímu místu
8	DisplayGatekeeperControlFcn	Překresluje aktuální informace o nabíjení

Tabulka 7: Funkce volané úlohou DispGatekpr

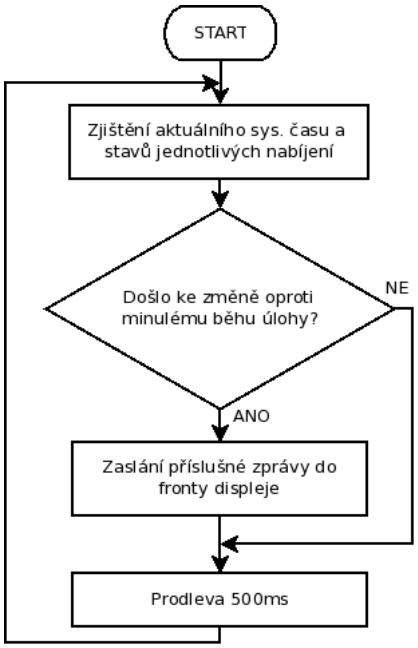
7.4.2 Aktualizace údajů na displeji

Úloha DispGatekpr sama o sobě nerozhoduje o tom, co a kdy na displej vykreslit. Přitom je např. potřeba periodicky překreslovat systémový čas či při výskytu určité události změnit obrázek ikony nebo tlačítka.

Tento problém je řešen úlohou DispUpdater, která je plánována periodicky (každých 500 ms), má nastavenou středně vysokou prioritu 3 a využívá stack paměť velikosti 100 B.

Jejím úkolem je v pravidelných intervalech zjišťovat systémový čas a v případě jeho změny zaslat jeho aktuální hodnotu do fronty úlohy DispGatekpr. Zároveň úloha zjišťuje, zda byl změněn stav nabíjení na některém odběrném místě. Dojde-li totiž k automatickému ukončení nabíjení (po nabití stanoveného množství energie), pak je opět potřeba zaslat zprávu do fronty.

Graficky lze mechanismus úlohy popsat vývojovým diagramem na obrázku 20.



Obrázek 20: Vývojový diagram úlohy DispUpdater

7.4.3 Vizualizace při čekání na identifikaci uživatele

Dalším členem skupiny úloh, které na displej přímo nepřistupují, přesto ale ovlivňují na něm zobrazené objekty, je DispCrdrdr. Tato úloha je plánována periodicky (každých 600 ms), má nízkou prioritu 1 a využívá paměť ve stacku o velikosti 150 B. Její důležitou vlastností je vazba na obrazovku pro identifikaci uživatele, na všech ostatních obrazovkách je úloha suspendována. Mechanismus je popsán vývojovým diagramem na obrázku 21.

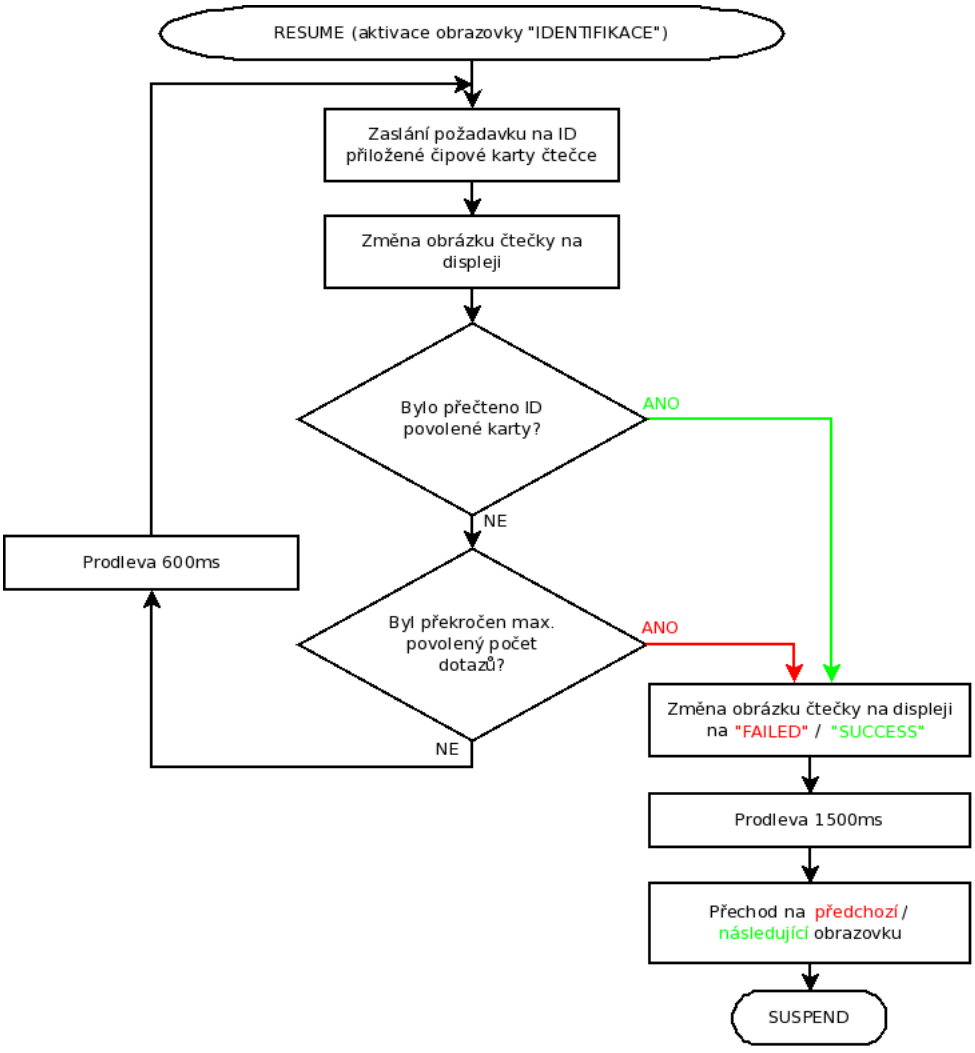
Po přechodu do stavu RUNNING úloha periodicky vykonává tři základní činnosti, dokud není splněna ukončovací podmínka. Postupně dochází k:

1. zaslání dotazu čtečce čipových karet na ID právě přiložené karty,
2. „animaci", tzn. změně ikony čtečky na displeji - prostřednictvím zaslání zprávy do fronty úlohy DispGatekpr (celkem jsou střídány 3 obrázky),

3. zjištění, zda je ke čtečce přiložena karta (tedy zda bylo vráceno některé z platných ID).

Poslední - třetí činnost - je zároveň první ukončovací podmínkou úlohy. Je-li totiž přečteno platné ID, dojde k jeho výpisu na displej a nastavení obrázku signalizujícího úspěch. Po prodlevě (600 ms) je pak úloha suspendována přepnutím na další obrazovku aplikace.

Zároveň se však může stát, že úloha DispCrdrdr je ve stavu RUNNING a zasílá dotazy čtečce karet, přitom ale žádná odpověď nepřichází (uživatel kartu nepřiloží). Proto musí být obsažena ještě druhá ukončovací podmínka: po projití určitého počtu (nastaveno 10) cyklů je signalizován neúspěch a úloha je opět po prodlevě 600 ms suspendována. V této situaci však nedochází k přepnutí na další obrazovku, ale na tu předchozí.



Obrázek 21: Vývojový diagram úlohy DispCrdrdr

7.4.4 Aktualizace informací o nabíjení

Poslední úlohou ovlivňující zobrazení na displeji je DispControl. Ta je obdobně jako DispCrdrdr vázána na konkrétní obrazovku aplikace - uplatňuje se pouze při monitoringu nabíjení. Úloha je vykonávána periodicky každé 2 s, její priorita je nastavena na nízkou hodnotu 2 a velikost pro ni alokovaného stacku je 100 B. Jejím úkolem je zasílání zprávy do fronty úlohy DispGatekpr. Následkem je zjištění informací o nabíjení aktuálního odběrného místa a jejich aktualizace na displeji.

7.5 Komunikace se čtečkou čipových karet

K identifikaci zákazníka, který chce nabíjecí stanici použít, je nutné komunikovat se čtečkou čipových karet ACR120. Ta je k řídicímu počítači připojena přes sériovou linku RS232.

7.5.1 Použití sériové linky

Nastavení komunikačního rozhraní je dáno možnostmi čtečky karet. Uvádí ho souhrně tabulka 8.

UART	Rychlost [Bd]	Datové bity	Stop bit	Parita
UART1	57600	8	1	žádná

Tabulka 8: Nastavení sériové linky pro komunikaci se čtečkou ACR120

Použit je binární protokol, přičemž všechny odesílané i přijímané telegramy mají formát dle tabulky 9.

STX	ID stanice	Délka příkazu	Příkaz / Data	BCC	ETX
1 B	1 B	1 B	N B	1 B	1 B

Tabulka 9: Formát telegramu čtečky ACR120

Chceme-li zjistit ID právě přiložené karty, je potřeba zaslat:

0x02 0x01 0x01 0x73 0x73 0x03,

kde jednotlivé byty popořadě jsou: STX=0x02, ID stanice=0x01, délka příkazu=0x01, příkaz=0x73 ('s' - select), BCC=0x73 a ETX=0x03.

Telegram, kterým čtečka odpovídá pak může mít tvar:

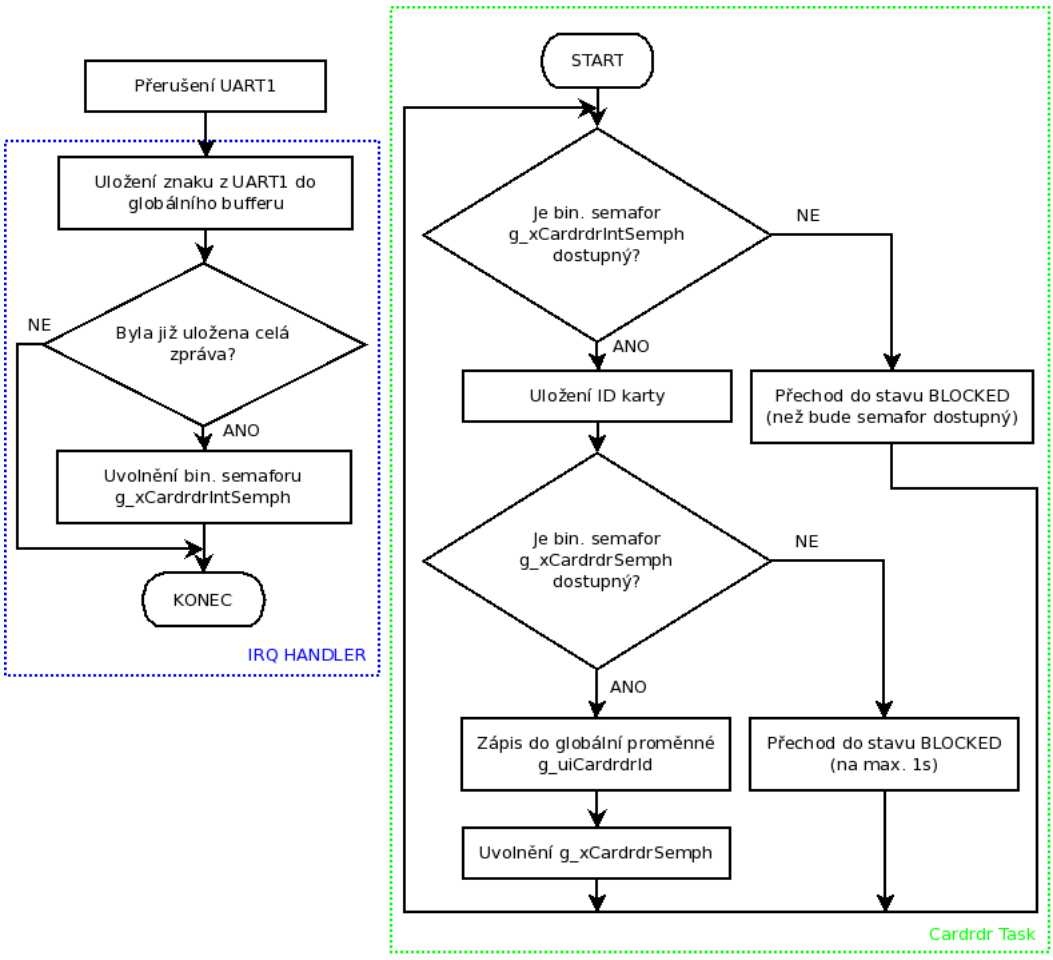
0x02 0x00 0x01 0x4e 0x4f 0x03,

kde data=0x4E značí nepřítomnost čipové karty. V případě, že je karta ke čtečce přiložena, je ve zprávě uloženo její ID o délce 4 B (označeno červeně):

0x02 0x00 0x04 0xNN 0xNN 0xNN 0xNN BCC 0x03.

7.5.2 Začlenění do programu

K odstartování komunikace se čtečkou obsahuje modul funkci, která je periodicky (každých 600 ms) volána z úlohy DispCardrdr (více v kapitole 7.4.3). Zpracování odpovědi je pak záležitostí rutiny obsluhující přerušení z UART1 a úlohy Cardrdr, která reaguje na událost, má vysokou prioritu 8 a využívá stack paměť o velikosti 50 B. Jejich součinnost je znázorněna vývojovými diagramy na obrázku 22.



Obrázek 22: Vývojové diagramy komunikace se čtečkou čipových karet

Poté, co je v ISR od čtečky obdržena celá zpráva, dochází k odblokování úlohy Cardrdr. Ta převede obdržený řetězec na celé číslo a v kritické sekci ho zapíše do globální proměnné *g_ uiCardrdrId*. Modul pak obsahuje funkci, kterou je tuto proměnnou možné bezpečným způsobem číst - volaná je z úlohy DispCardrdr.

7.6 Komunikace s elektroměry

K měření odběru energie jsou určeny tři elektroměry ED 310.DR, se kterými lze komunikovat po sériové lince RS485.

7.6.1 Nastavení RS485

Komunikace s elektroměry je trochu složitější, protože je potřeba přepínat směr toku. To je prováděno volným pinem PB6 vývojového kitu DK-LM3S9B96. Nastavení sériové linky je pak uvedeno v tabulce 10.

UART	Rychlost [Bd]	Datové bity	Stop bit	Parita
UART0	2400	7	1	sudá

Tabulka 10: Nastavení sériové linky pro komunikaci s elektroměry

Na rozdíl od čtečky je s elektroměry komunikováno ASCII protokolem. Každé ze tří měřicích zařízení je specifikováno svou adresou. Zaslání požadavku na informace o nabíjení má proto tvar:

/ ? Adresa ! CR LF.

Odpověď je poměrně obsáhlá, jak je znázorněno na obrázku 23.

```
/ZPA4ZE310.v30_007/r/n          // Typ elektromeru
<STX>                           // Start
F.F(000000)/r/n                 // ID chyby
C.90(826371)/r/n                // Komunikacni adresa
1.8.1(0000003.8#kWh)/r/n        // Celkovy odber na vsech fazich
21.8.0(0000001.5*kWh)/r/n       // Celkovy odber na fazi L1
41.8.0(0000001.5*kWh)/r/n       // Celkovy odber na fazi L2
61.8.0(0000001.5*kWh)/r/n       // Celkovy odber na fazi L3
31.7(000.000*A)/r/n             // Okamzita ef. hodnota proudu na L1
51.7(000.000*A)/r/n             // Okamzita ef. hodnota proudu na L2
71.7(000.000*A)/r/n             // Okamzita ef. hodnota proudu na L3
1.6.1(00.0000*kW)/r/n           // Okamzity cinny vykon na L1
1.6.2(00.0000*kW)/r/n           // Okamzity cinny vykon na L2
1.6.3(00.0000*kW)/r/n           // Okamzity cinny vykon na L3
!/r/n<ETX>                      // Ukonceni
<BCC>                           // Kontrolni soucet
```

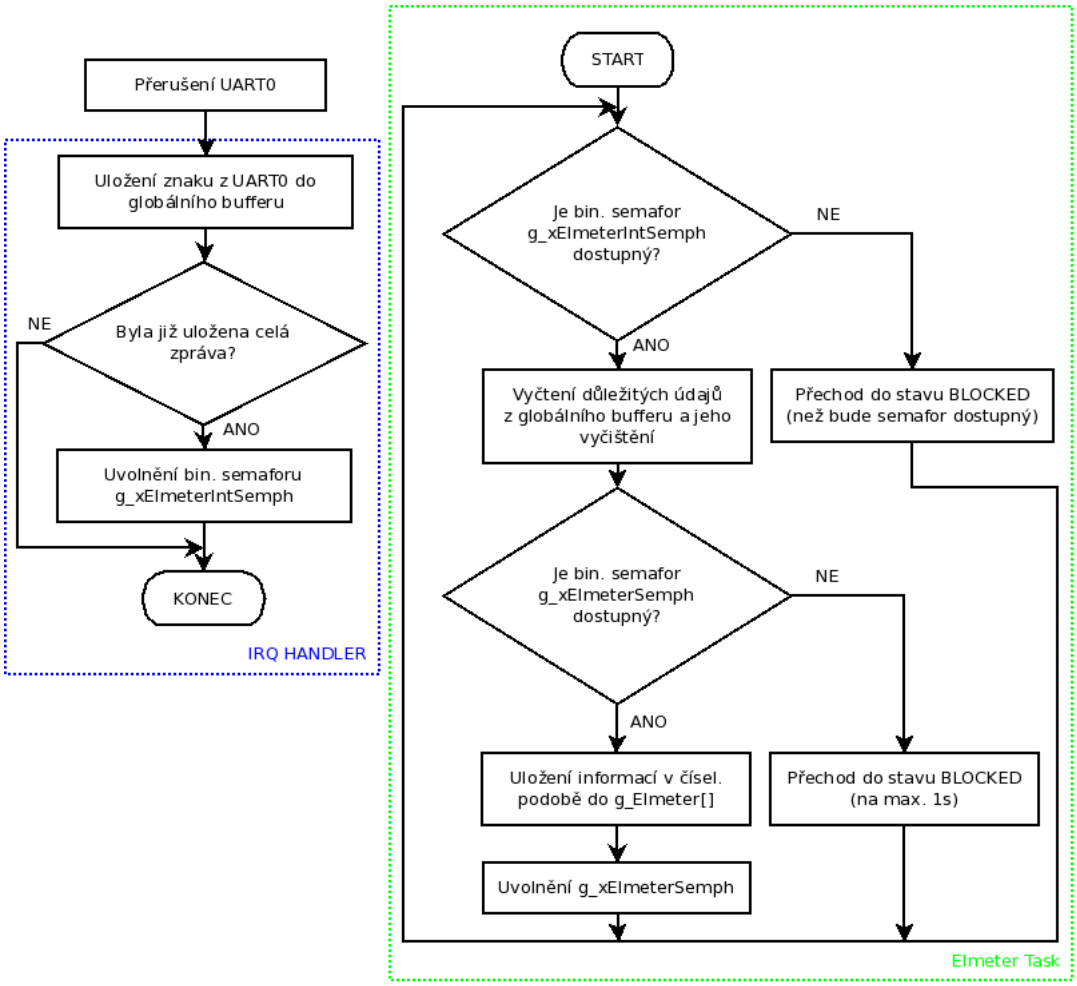
Obrázek 23: Zpráva od elekroměru

Uvedena je samozřejmě adresa zařízení, celkový odběr i odběr na jednotlivých fázích, dále pak okamžitá efektivní hodnota proudu a činného výkonu. Důležitá data jsou opět ohraničena byty STX a ETX.

7.6.2 Začlenění do programu

Komunikaci s jednotlivými elektroměry začíná vždy úloha Charging, jak bude dále uvedeno v kapitole 7.8. Zde bude popsáno zpracování balíku přijatých dat, které je řešeno úlohou Elmeter. Ta reaguje na událost (když je přijat celý telegram od příslušného elektroměru), má vysokou prioritu 7 a využívá 150 B paměti ve stacku.

Postup zpracování dat popisují vývojové diagramy rutiny obsluhující přerušení z UART0 a úlohy Elmeter na obrázku 24.



Obrázek 24: Vývojové diagramy zpracování dat z elektroměrů

Z diagramů je patrné, že většina práce je provedena už v ISR, kde jsou postupně ukládány znaky přijaté od elektroměrů do bufferu *g_ucElmeterBuff*. Je-li přijatá zpráva kompletní, pak je v ISR uvolněn semafor *g_xElmeterIntSemph*.

Následkem je odblokování úlohy Elmeter, která má za úkol přijatý buffer zpracovat - tj. vyčíst z něj parsováním důležité informace. Ty jsou převedeny do číselné

podoby a následně (v semaforem *g_xElmeterSemph* chráněné kritické sekci) uloženy do globální proměnné *g_Elmeter[]*. Součástí modulu je pak samozřejmě funkce, kterou lze k těmto číselným údajům bezpečně přistupovat. Volána je z úlohy Charging.

7.7 Ovládání stykačů

Tento modul neobsahuje žádnou úlohu, ale pouze funkci, kterou lze přistupovat k pinům mikrokontroléru LM3S9B96, jimiž lze ovládat stykače. Konkrétní přiřazení uvádí tabulka 11.

Odběrné místo	Konfigurace	Pin
1	400 V / 63 A	PB2
2	400 V / 32 A	PB3
3	230 V / 16 A	PF1

Tabulka 11: Piny určené k ovládání stykačů

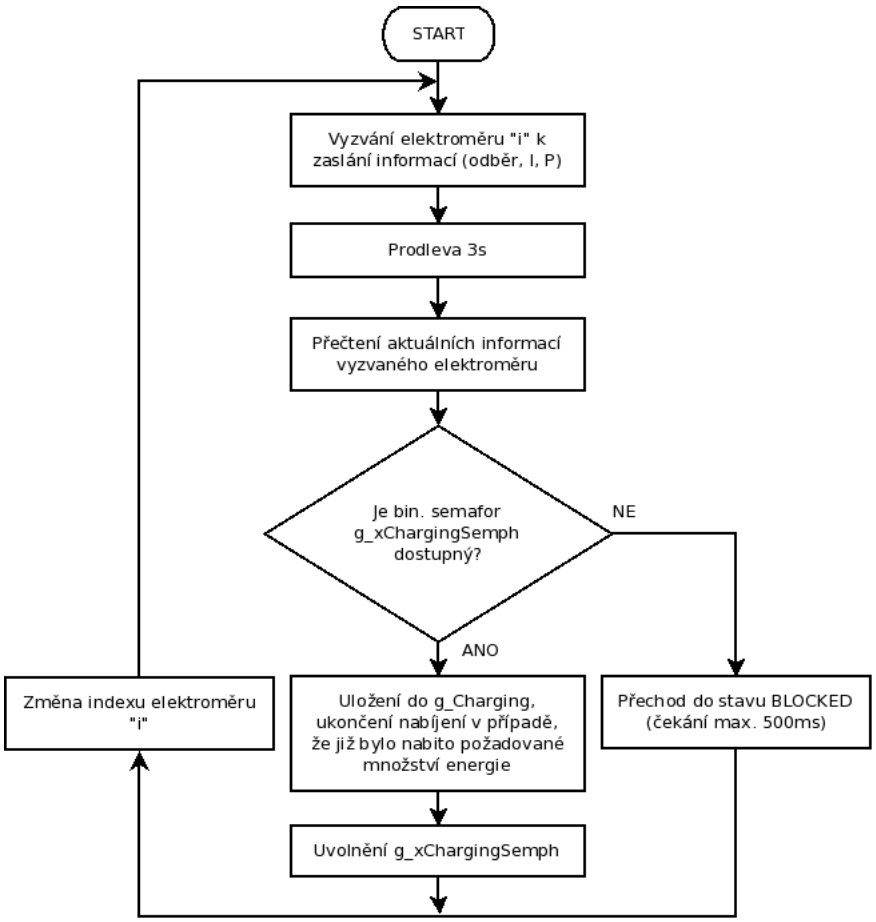
Funkce pro ovládání stykačů je volána z úlohy Charging.

7.8 Kontrola průběhu nabíjení

Tento důležitý modul je určen k monitoringu probíhajících nabíjení a k jejich případnému ukončení. Využívá proto informace čtené z elektroměrů a stejně tak má přístup k ovládání GPIO stykačů jednotlivých odběrných míst. Informace o nabíjení zároveň zprostředkovává úloze DispControl, která je může vypisovat na displej.

Výše uvedené činnosti jsou prováděny úlohou Charging, která je plánována periodicky (každé 3 s), má středně vysokou prioritu 4 a využívá stack paměť velikosti 100 B. Její vývojový diagram je na obrázku 25.

Jak je z něj patrné, úloha v nekonečném cyklu vyzývá postupně jednotlivé elektroměry k zaslání jimi uchovávaných informací (odběr, okamžitá efektivní hodnota proudu a činný výkon). Po prodlevě 3 s se snaží tyto informace přečíst a uložit do struktury g_Charging[], která krom aktuálních údajů z elektroměrů uchovává např. stav odběru před začátkem nabíjení či požadovaný odběr. Díky těmto informacím pak lze rozhodnout i o případném ukončení nabíjení.



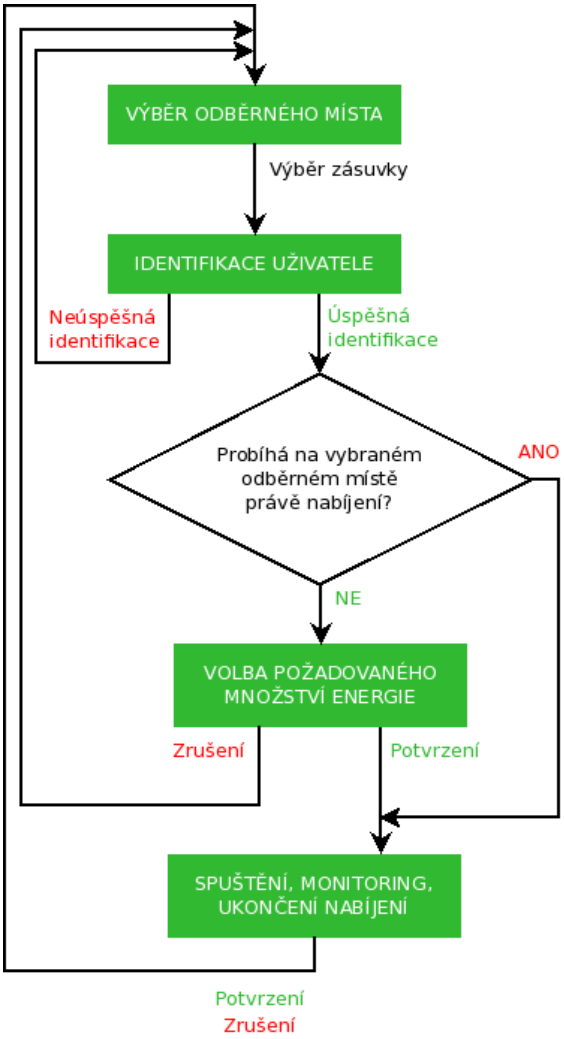
Obrázek 25: Vývojový diagram úlohy Charging

8 ŘÍDICÍ APLIKACE - OVLÁDÁNÍ

K ovládání řídicí aplikace je určen dotykový displej, na němž je zobrazeno jednoduché grafické uživatelské rozhraní. Jeho základní logika bude popsána v této kapitole.

8.1 Obrazovky a přepínání mezi nimi

Uživatelské rozhraní je členěno na čtyři základní obrazovky, mezi kterými lze přepínat dle schématu znázorněného na obrázku 26 (pozn.: nejedná se o vývojový diagram).



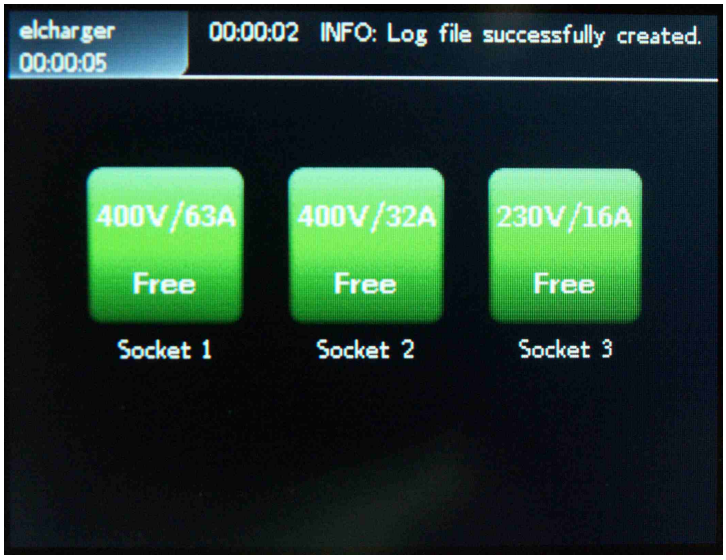
Obrázek 26: Schéma přepínání mezi obrazovkami

Zelené bloky zde odpovídají jednotlivým obrazovkám. K přepnutí z té úvodní („Výběr odběrného místa") je nutné vybrat jednu ze zásuvek. Druhá pak je zob-

razena pouze omezený čas - maximálně 6 s. Směr přechodu závisí na úspěšnosti identifikace (tzn. zda uživatel přiloží platnou čipovou kartu ve stanoveném časovém limitu) a na tom, zda na vybraném odběrném místě právě probíhá nabíjení či ne. Zbylé dvě obrazovky obsahují tlačítka typu „Potvrdit" a „Zrušit", kterými je určen směr přepnutí.

8.2 Výběr odběrného místa

Úvodní obrazovka, která nabíhá po startu programu, je znázorněna na obrázku 27.



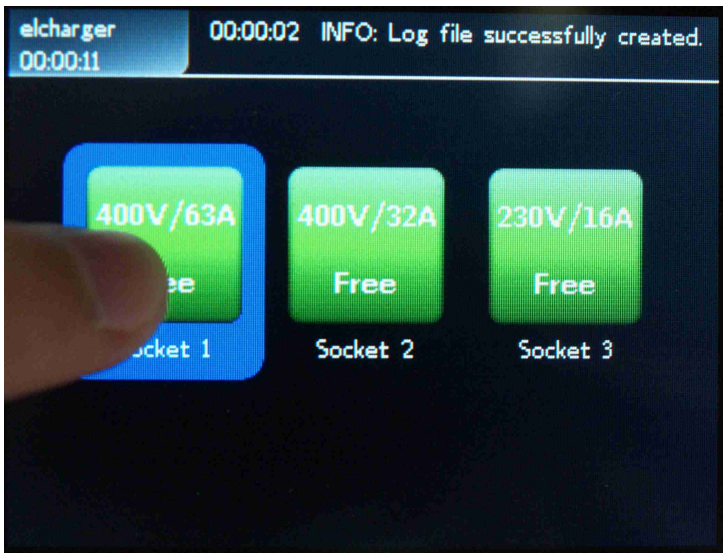
Obrázek 27: Úvodní obrazovka aplikace - volná všechna nabíjecí místa

V její horní části je umístěno záhlaví tvořené jménem aplikace, aktuálním systémovým časem a dvouřádkovým logem, ve kterém jsou vypsány poslední informační či chybové zprávy. Toto záhlaví je společné všem obrazovkám.

Hlavní obsah úvodní obrazovky však tvoří tři ikony, z nichž každá přísluší vždy jednomu odběrnému místu. Jednotlivé zásuvky mohou být označeny jako:

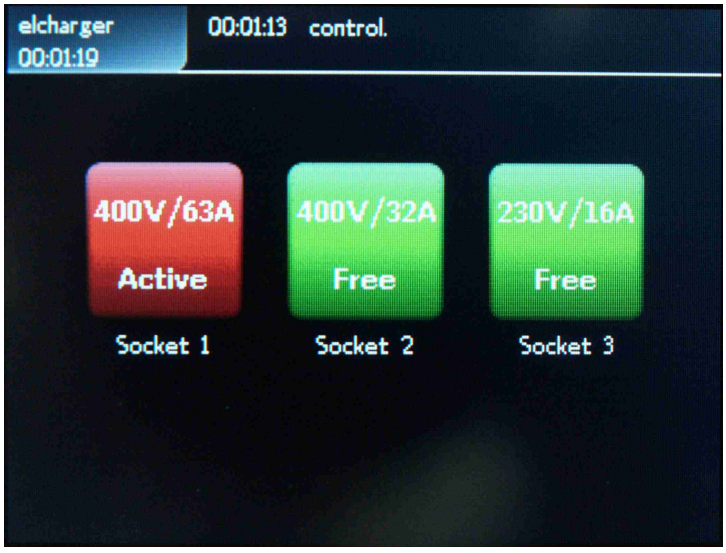
- Free nebo
- Active.

Na volných zásuvkách (Free) žádné nabíjení v daném okamžiku neprobíhá, proto mohou být aktivovány libovolným uživatelem, jak je znázorněno obrázkem 28.



Obrázek 28: Úvodní obrazovka aplikace - volba nabíjecího místa

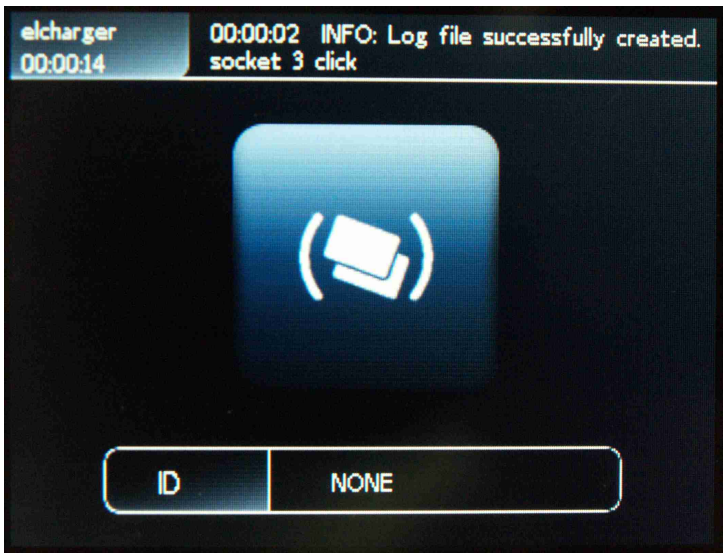
Je-li nabíjecí místo aktivní, tzn. probíhá-li na něm právě nabíjení, vypadá úvodní obrazovka podle obrázku 29.



Obrázek 29: Úvodní obrazovka aplikace - obsazené nabíjecí místo

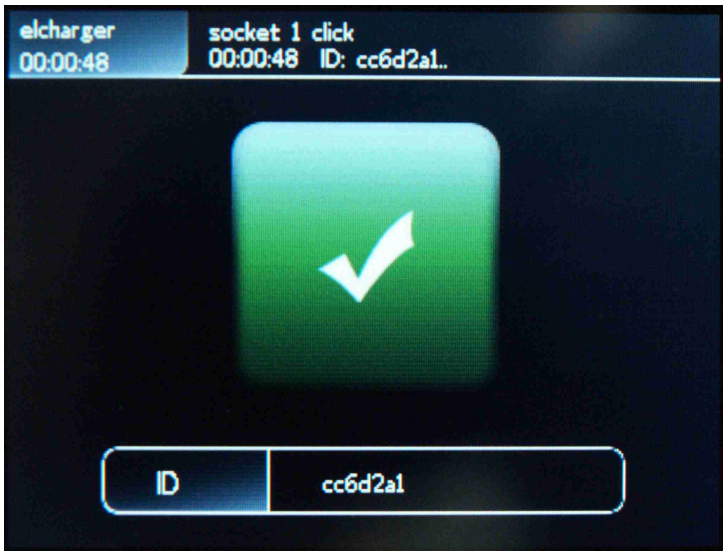
8.3 Identifikace uživatele

Po výběru jednoho ze tří dostupných odběrných míst stanice je uživatel vyzván k identifikaci čipovou kartou (viz. obrázek 30).



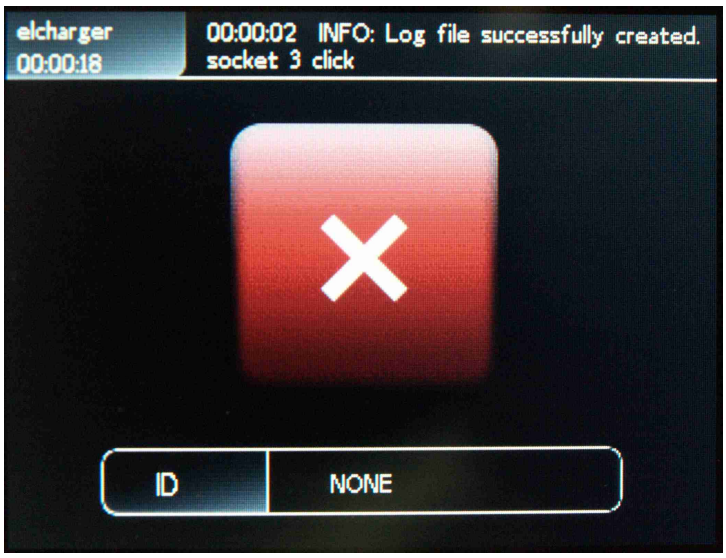
Obrázek 30: Čekání na přiložení čipové karty

Přiloží-li uživatel ke čtečce platnou kartu a je-li ID této karty systémem úspěšně přečteno, pak je signalizován úspěch (viz. obrázek 31). Následně dojde po krátké prodlevě (1,5 s) k automatickému přepnutí na další obrazovku, která je určena v závislosti na tom, zda na vybraném odběrném místě právě probíhá nabíjení či nikoliv (viz. schéma na obr. 26).



Obrázek 31: Úspěšná identifikace

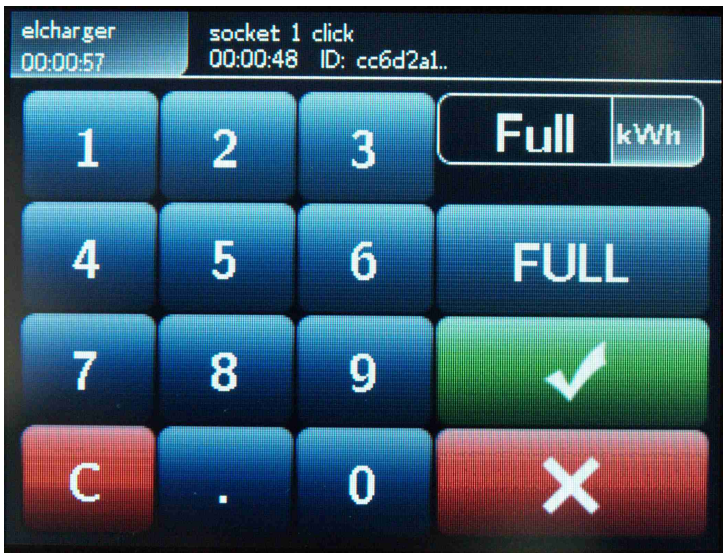
Nestihne-li uživatel ve stanoveném časovém limitu (6 s) kartu přiložit, pak je signalizován neúspěch (viz. obrázek 32). Obdobným způsobem systém odmítne i zákazníka, který se chce přihlásit svou kartou k místu, které je již obsazené někým jiným. V obou případech je následně přepnuto zpět na úvodní obrazovku.



Obrázek 32: Neúspěšná identifikace

8.4 Volba požadovaného množství energie

Uživatel, který si chce aktivovat volné nabíjecí místo, je po úspěšné identifikaci vyzván k nastavení požadovaného odběru (viz. obrázek 33).



Obrázek 33: Nastavení nabíjení

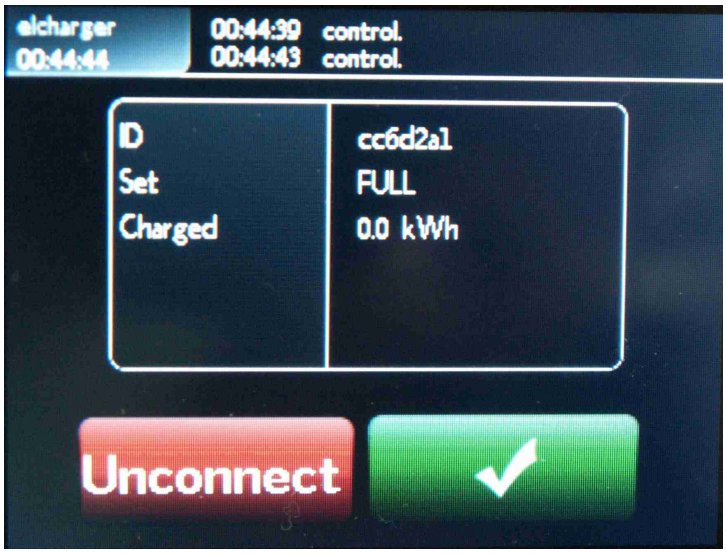
Dostupná jsou dvě různá nastavení: plné nabití baterie „Full" (implicitní) či specifikace číselnou hodnotou v kWh. Obrazovku lze opustit potvrzením nastavení nebo jeho zrušením, což způsobí přechod na úvodní obrazovku.

8.5 Spuštění nabíjení a monitoring

Po zvolení parametrů nabíjení doposud volného odběrného místa je zákazník vyzván ke kontrole údajů (viz. obrázek 34). Pokud je s nastavením spokojen, může spustit nabíjení. V opačném případě lze start procesu zrušit.

Obrázek 34: Spuštění nabíjení

Trochu jiný vzhled má obrazovka v případě, že na zvoleném odběrném místě již nabíjení probíhá (viz. obrázek 35). Zákazník zde může zjistit, kolik z požadované energie již bylo nabito, a případně nabíjení ukončit.

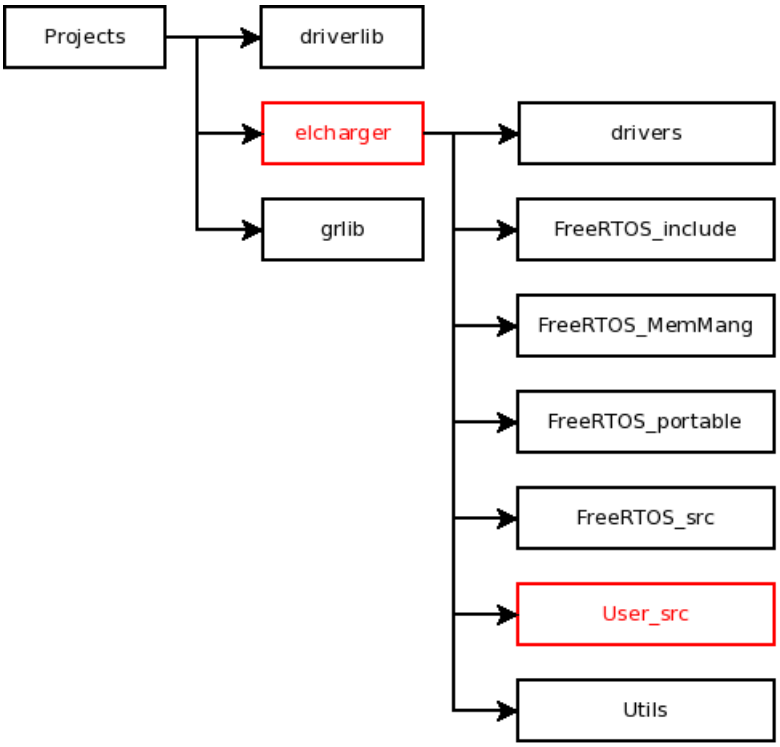


Obrázek 35: Monitoring nabíjení

9 ŘÍDICÍ APLIKACE - ZDROJOVÉ MODULY

Náplní této kapitoly je přehled zdrojových modulů programu a popis jejich rozmístění v souborovém systému.

Topologii složek projektu znázorňuje obrázek 36.



Obrázek 36: Struktura zdrojových souborů

Zdrojové soubory obsahující vlastní kód programu elcharger jsou umístěny pouze ve složce User_src. Ostatní složky obsahují operační systém FreeRTOS a podpůrné knihovny dodávané firmou Texas Instruments.

9.1 Knihovna Driverlib

Jedná se o knihovnu, která je dodávána spolu s vývojovými kity od společnosti Texas Instruments. Obsahuje makra a funkce pro přístup k periferiím mikrokontrolérů řady LM3S*.

9.2 Knihovna Grlib

Grafická knihovna Grlib je, podobně jako Driverlib, součástí SW podpory pro vývojové kity Texas Instruments. Použít lze jak pro jednoduché vypisování či vy-

kreslování na displej, tak pro tvorbu uživatelského rozhraní s tlačítky, textovými poli, apod.

9.3 Firmware DK-LM3S9B96

Ve složce elcharger/drivers je umístěna speciální skupina zdrojových souborů dodávaných společně s kitem DK-LM3S9B96. Podporována je např. práce s externími RAM a FLASH paměťmi či displejem, tedy specifickými komponentami kitu.

9.4 Operační systém FreeRTOS

Zdrojové soubory OS FreeRTOS, dostupné ke stažení např. z [6], zahrnují kód pro práci s úlohami, frontami či semafory. Dále obsahují makra a funkce specifické pro rodinu mikrokontrolerů ARM Cortex-M3.

Při vývoji projektu je používána verze systému 6.0.2.

9.5 Aplikační kód

Veškeré nově vytvořené zdrojové soubory, jimiž je definováno chování programu elcharger, jsou uloženy ve složce User_src. Jsou to:

main.c: Zde je obsažena v podstatě pouze funkce main( ), ve které jsou inicializovány (a tím zároveň spuštěny) jednotlivé moduly.

cardreader.c, cardreader.h: Obsahují funkce pro zjišťování ID karty přiložené ke čtečce.

contactor.c, contactor.h: V těchto souborech jsou implementovány funkce pro přístup k GPIO, ke kterým jsou připojeny stykače.

cr_startup.c: Obsahuje startovací skripty, dále je zde možné přiřadit obslužné rutiny jednotlivým typům přerušení.

display.c, display.h: Zde je uložen kód, který přímo souvisí s vykreslováním na displej, tzn.: úlohy, inicializační funkce jednotlivých obrazovek, odezvy kliknutí na tlačítka grafického rozhraní, atd.

elmeter.c, elmeter.h: Soubory zahrnují funkce pro přijímání a zpracování telegramů elektroměrů.

FreeRTOSConfig.h: Zde je možné nastavit některé vlastnosti a chování OS FreeRTOS - např. počet úrovní priorit či možnost použití některých funkcí.

graphics.c, graphics.h: V těchto souborech je implementováno rozhraní pro základní přístup k displeji - výpis řetězce, vykreslení čáry, apod.

charging.c, charging.h: V souborech charging.* jsou obsaženy funkce pro monitoring průběhu nabíjení.

log.c, log.h: Jak název napovídá, je zde implementováno rozhraní pro logování zpráv.

rt.c, rt.h: Obsahují modul pro udržování reálného (systémového) času aplikace.

srdcard.c, sdcard.h: V těchto souborech je zahrnuto rozhraní pro čtení a zápis na souborový systém na SD kartě.

uart.c, uart.h: Obecné funkce pro práci s rozhraním UART. Používány jsou jak pro komunikaci s čtečkou čipových karet, tak s elektroměry.

widget.c, widget.h: V těchto souborech jsou obsaženy definice objektů vykreslovaných na displej a samozřejmě funkce, kterými k nim lze přistupovat.

9.6 Zdrojové soubory třetích stran

Ve složce elcharger/Utils je uložen software třetích stran, který je k dispozici také od firmy Texas Instruments. Ačkoli není jejím majetkem, je uvolněn pod otevřenou licencí a může být používán v nových projektech. Obsažen je v následujících zdrojových souborech:

ustdlib.c, ustdlib.h: Tyto soubory obsahují funkce pro formátování řetězců, které tvoří náhradu klasických funkcí ze souboru <string.h>.

diskio.h, ff.c, ff.h, integer.h, mmc-dk-lm3s9b96.c Uvedené soubory je nutné začlenit do projektu, chceme-li využívat souborový systém FAT např. na FLASH paměti.

10 ZÁVĚR

V rámci diplomové práce byl nejprve vypracován přehled trhu bezobslužných nabíjecích stanic pro elektromobily. Poměrně patrné je, že ačkoli na trhu několik výrobků existuje, jedná se většinou spíše o prototypy, které je třeba do budoucna rozvíjet. To není příliš překvapivou informací vzhledem k nynějšímu malému rozšíření elektromobilů.

Dalším zpracovaným tématem bylo vytvoření uceleného přehledu o prototypu nabíjecí stanice společnosti ELNICO s.r.o. V textu je uveden popis jednotlivých komponent, jejich propojení je pak znázorněno blokovým schématem. Samostatně je popsán také řídicí počítač realizovaný vývojovým kitem DK-LM3S9B96 a operační systém FreeRTOS, jímž je vybaven.

Hlavní náplní práce byla tvorba softwarového vybavení pro výše uvedený řídicí počítač tak, aby mohl být použit k ovládání nabíjecí stanice. Díky použití operačního systému reálného času bylo možné celý problém rozčlenit na několik samostatně funkčních úloh. Snahou také bylo, aby byl program co nejvíce modulární a jeho jednotlivé části při dodržení určitých pravidel bylo možné použít odděleně.

K ovládání aplikace bylo vytvořeno jednoduché grafické rozhraní. V něm si uživatel může na úvodní obrazovce vybrat jedno ze tří dostupných odběrných míst. Následná identifikace probíhá přiložením čipové karty ke čtečce, s níž je komunikováno po sériové lince RS232. Po nastavení požadovaného odběru energie je již možné spustit nabíjecí proces. K tomu jsou určeny stykače, ovladatelné prostřednictvím GPIO pinů mikrokontroléru. Zpětnou vazbu z nabíjecího procesu získává aplikace vyčítáním informací (především aktuálního odběru) z připojených elektroměrů po lince RS485. Nabíjení lze kdykoli (po opětovné identifikaci uživatele) přerušit, implementováno je navíc automatické ukončení po nabití požadovaného množství energie.

Pro praktické využití navrženého SW by bylo vhodné nejprve provést některá vylepšení a rozšíření. Současná implementace např. příliš neřeší kontrolu konzistence dat přijatých z elektroměrů. Při vývoji tohoto modulu také vzniklo největší zpoždění tím, že elektroměry byly firmou ZPA chybně naprogramovány a komunikace s nimi v podstatě nefungovala. Tento problém musel být řešen reklamací. Současný stav však stále není úplně uspokojující, protože elektroměry komunikují rychlostí výrazně pod svůj potenciál - přijetí telegramu trvá déle než 2 s. Tento čas by pro danou délku zprávy a nastavenou baudovou rychlost mohl být kratší než 1 s. Problémem však zůstává, že elektroměr neodpovídá na zprávu, jejímž za-

sláním by podle dokumentace mělo být možné odpověď urychlit. Tato záležitost bude pravděpodobně muset být řešena další reklamací.

11 LITERATURA

[1] *ACR120 Contactless Reader/Writer - Technical Specifications* [online]. Advanced Card Systems Limited. Poslední revize červen 2008 [cit. 2009-12-04]. Dostupné z WWW: <http://www.acs.com.hk/index.php?pid=product&id=ACR120>.

[2] *ACR120 Contactless Reader - A Product Presentation* [online]. Advanced Card Systems Limited. [cit. 2009-12-04]. Dostupné z WWW: <http://www.acs.com.hk/index.php?pid=product&id=ACR120>.

[3] *ACR120 Contactless Reader/Writer - Communication Protocol* [online]. Advanced Card Systems Limited. [cit. 2009-12-04]. Dostupné z WWW: <http://www.acs.com.hk/index.php?pid=product&id=ACR120>.

[4] Barry, R. *FreeRTOS Reference Manual - API Functions and Configuration Options* [online]. [cit. 2009-11-30]. Dostupné z WWW: <http://www.freertos.org/Documentation/FreeRTOS-documentation-and-book.html>.

[5] Barry, R. *Using The FreeRTOS Real Time Kernel - Practical Guide* [online]. [cit. 2009-11-29]. Dostupné z WWW: <http://www.freertos.org/Documentation/FreeRTOS-documentation-and-book.html>.

[6] *FreeRTOS Real Time Kernel by Richard Barry* [online]. SourceForge. c2009 [cit. 2009-12-10]. Dostupné z WWW: <http://sourceforge.net/projects/freertos/files/FreeRTOS/>.

[7] *GRT508D-100A* [online]. Great Electrical. c2009 [cit. 2009-12-02]. Dostupný z WWW: <http://www.great-relay.com/img/tpl/pdf/508D-100A-e.pdf>.

[8] *Heccharger* [online]. Hectronic. [cit. 2009-11-14]. Dostupné z WWW: <http://www.hectronic.de/en/news-presse/newsview/news/pionierstimmung-auf-der-ecartec-in-muenchen/?tx_ttnews[day]=20&tx_ttnews[month]=10&tx_ttnews[year]=2009&cHash=0c3337cd91>.

[9] Horčík, J. *Hybrid.cz* [online]. 8.3.2010 [cit. 2010-04-12]. ČEZ má dodavatele prvních dobíjecích stanic pro elektromobily. Dostupné z WWW: <http://www.hybrid.cz/novinky/cez-ma-dodavatele-prvnich-dobijecich-stanic-pro-elektromobily>.

[10] *LM3S9B96 Microcontroller Datasheet* [online]. Texas Instruments. Poslední revize 9.10.2009 [cit. 2009-11-22]. Dostupné z WWW: <http://www.luminarymicro.com/products/lm3s9b96.html>.

[11] *Odečty elektroměrů dle ČSN EN 62056-21.* ZPA Smart Energy a.s. 3 s.

[12] *Optimizing Code Performance and Size for Stellaris® Microcontrollers.* Luminary Micro. c2007. 20 s.

[13] *P.CHARGE* [online]. Schletter. [cit. 2009-11-14]. Dostupné z WWW: <http://www.schletter.de/cs/component/content/article/125/139.html>.

[14] *Programming the On-Chip Flash Memory in a Stellaris® Microcontroller.* Luminary Micro. c2007. 11 s.

[15] *Rittal - eMobility Charging Stations of the Future* [online]. Rittal. [cit. 2010-04-12]. Dostupné z WWW: <http://www.rittal.com/products/communication-systems/technology/index_elektro.html#headline1>.

[16] *Stellaris Graphics Library - User's Guide* [online]. Texas Instruments. Poslední revize 1.10.2009 [cit. 2009-12-09]. Dostupné z WWW: <http://www.luminarymicro.com/products/software_updates.html>.

[17] *Stellaris LM3S9B96 Development Kit, User's Manual* [online]. Texas Instruments. Poslední revize 29.6.2009 [cit. 2009-11-17]. Dostupné z WWW: <http://www.luminarymicro.com/products/dk-lm3s9b96.html>.

[18] *Stellaris Peripheral Driver Library - User's Guide* [online]. Texas Instruments. Poslední revize 1.10.2009 [cit. 2009-12-07]. Dostupné z WWW: <http://www.luminarymicro.com/products/software_updates.html>.

[19] *The CITEA Family* [online]. Hectronic. c2009 [cit. 2009-11-14]. Dostupné z WWW: <http://www.hectronic.de/en/products/parking/the-citea-family/>.

[20] *The FreeRTOS Project* [online]. 16.10.2009 [cit. 2009-12-06]. Dostupný z WWW: <http://www.freertos.org/>.

[21] *The new charging station complEo.* EBG. [online]. [cit. 2009-11-14]. Dostupné z WWW: <http://ebg-luenen.de/ebgluenen_en/produkte/08_charging_station_compleo.php>.

[22] *Třífázové statické činné elektroměry ED 310, ED 310.I* [online]. ZPA. Poslední revize březen 2009 [cit. 2009-12-01]. Dostupné z WWW: <http://www.zpa.cz/index.php/cz/produkty_a_reseni__1/elektromery/trifazove_na_din/ed_310>.

12 SEZNAM ZKRATEK

ADC	Analog to Digital Converter
AES	Advanced Encryption Standard
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
BCC	Block Check Character
CAN	Controller Area Network
CPLD	Complex Programmable Logic Device
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
DC	Direct Current
DPS	Deska plošného spoje
EPI	External Peripheral Interface
ETX	End of Text
FAT	File Allocation Table
FIFO	First In First Out
FPGA	Field-Programmable Gate Array
GND	Ground
GPIO	General Purpose Input/Output
GPRS	General Packet Radio Service
GSM	Global System for Mobile communications
GPL	General Public License
HMI	Human Machine Interface
HW	Hardware

Hz	Hertz
I2S	Inter-IC Sound
ICDI	In-circuit Debug Interface
I/O	Input/Output
ISR	Interrupt Service Routine
LCD	Liquid Crystal Display
LED	Light-Emitting Diode
MAC	Media Access Controller
MUTEX	Mutual Exclusion
OS	Operační systém
PC	Personal Computer
PHY	Physical Interface
RAM	Random-Access Memory
RFID	Radio Frequency Identification
RGB	Red Green Blue
ROM	Read-Only Memory
RS232	Recommended Standard 232
RS485	Recommended Standard 485
RTOS	Real Time Operating System
SD	Secure Digital
SMS	Short Message Service
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
SSI	Synchronous Serial Interface

STX	Start of Text
SW	Software
TCP/IP	Transmission Control Protocol / Internet Protocol
TTL	Transistor-Transistor Logic
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
USB OTG	Universal Serial Bus On-The-Go